Angelo Burgener

Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Bachelorarbeit 2013

Studiengang Wirtschaftsinformatik

digital library

Abbildung 1: Digital library Bild (ICTS, Jahr unbekannt)

Student : Angelo / Burgener

Dozent : Müller / Henning

Abgabetermin : 29. Juli 2013

I Vorwort

Damit die Bachelorarbeiten gerecht verteilt werden konnten, musste zuerst eine Selbstbeurteilung vorgenommen werden. Hier hatte jeder Student die Möglichkeit seine Stärken und Schwächen anzugeben. Diese Angaben wurden bei der Verteilung der Arbeiten berücksichtigt.

Am 03.04.2013 wurden die vorgeschlagenen Arbeiten im Intranet freigeschaltet. Jetzt musste eine Priorität von 6 Aufgaben erstellt werden. Die Wahl der Prioritäten war für mich nicht so einfach, da ich mir noch nicht richtig vorstellen konnte, was in welchem Projekt genau verlangt wird. Jede Arbeit hatte seine eigenen speziellen Neuheiten. Somit habe ich die Schwierigkeiten ausprobiert und mich gefragt, wie interessiert mich das Thema und welche Kompetenzen habe ich, diese Arbeit auch gut zu bewältigen. Schlussendlich habe ich mich für die digital library entschieden. Der Vorteil hier war, dass ich eine Web Applikation schreiben kann und hierfür auch PHP (Hypertext Preprocessor) verwenden darf.

Meine Motivation zu dieser Arbeit liegt darin, dass ich Neues mit schon Bekanntem vermischen kann. Da ich für die Webapplikation PHP verwenden kann, muss ich mich in diesem Bereich nicht mehr so stark informieren. Hingegen Apache Solr/ Tika ist für mich absolut neu. Diese Elemente richtig zu verstehen wird wohl meine Hauptaufgabe werden. So kann ich zum bereits bestehenden Wissen Neues hinzufügen, ohne dass die ganze Arbeit neu sein muss und somit auch mehr Zeit für die neuen Elemente aufbringen.

Das größte Problem wird sicherlich sein, die neuen Elemente in meine PHP Anwendung einzubinden, da diese JAVA basiert sind. Dadurch werde ich auch für PHP neue Elemente kennen lernen.

Ich freue mich auf diese Arbeit und hoffe, dass ich sie auch wunschgemäß abschließen werde.

II Inhaltsverzeichnis

III Tabellenverzeichnis

IV Abbildungsverzeichnis

V Abkürzungsverzeichnis

PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
XML	Extensible Markup Language
CSS	Cascading Style Sheets
PDF	Portable Document Format
GUI	Graphical User Interface
ASF	Apache Software Foundation
FOSS	Free and Open Source Software
HTTPD	Hypertext Transfer Protocol Daemon (Server)
API	Application programming Interface
JS	Java Script
URL	Uniform Resource Locator
SQL	Structured Query Language
JAR	Java Archive
SMTP	Simple Mail Transfer Protocol
TXT	Textdatei

Einleitung

Das eigentliche Ziel der Arbeit ist eine digitale Bibliothek zu erstellen. Um dies realisieren zu können, wird ein Webservice (Apache SolR) zur Verfügung gestellt. Mit diesem Service wird es möglich sein, verschiedene Dokumente, in dieser Arbeit vor allem PDF's, zu indizieren. So werden verschiedene Optionen für die Suche ermöglicht. Diese Anwendung wird als Web-Applikation verlangt, einerseits für den Desktop, anderseits für Mobile. Sie wird in PHP geschrieben. Zusätzlich muss ein social network für diese Dokumente erstellt werden, das heißt, für alle Dokumente kann der User seine Bewertung abgeben. Dies erfolgt über eine Eingabe von Sternen, z.B. kein Stern bedeutet, ist unbrauchbar, ein Stern ist brauchbar, zwei Sterne ist gut, drei Sterne ist sehr gut, vier Sterne ist hervorragend und fünf Sterne ist ausgezeichnet. So kann der User die verschiedenen Dokumente bewerten. Außerdem hat er die Möglichkeit, zusätzlich einen Kommentar hinzuzufügen. Dieser wird dann von allen Benutzern gesehen und kann auch wiederrum kommentiert werden.

Apache SolR, wie in Kapitel 2.3 beschrieben, arbeitet ausschließlich mit XML Dateien. Damit dieser Service auch PDF's einlesen kann, wird ein zusätzlicher Service verwendet. Dieser hat den Namen Apache Tika. Mit dessen Hilfe können Metadaten und Inhaltsinformationen extrahiert werden. Genaueres finden Sie im Kapitel 2.4. Mit diesen wird dann ein XML File erstellt, welches vom Apache SolR gelesen werden kann und somit indiziert wird. Außerdem werden die Bilder des PDF's in einen separaten Ordner gespeichert. Hierfür wird der Service PDFBox verwendet. Dieser wird im Kapitel 2.5 beschrieben.

Damit der User die Webapplikation verwenden kann muss er ein Account haben. Dieser kann nur der Admin erstellen. Alle zusätzlich erstellten Profile haben diese Funktion nicht. Wenn ein Account existiert, kann der User mit einem Passwort sich einloggen. Anschließend hat er Zugriff auf alle Funktionen der Applikation. Falls er nun PDF's indizieren möchte, kann er die gewünschten Dateien in einen spezifisch für den User erstellten Ordner auf dem FTP-Server (medgift.hevs.ch) kopieren. Anschließend müssen in der Applikation verschiedene Informationen, wie Jahr, Abkürzung der Konferenz, die Konferenz ausgeschrieben, Land und Ort eingegeben werden. Erst dann können die Dateien, durch einen Button, indiziert werden. Jetzt arbeitet die Applikation. Damit der User weiß wie lange er noch warten muss, wird ein Ladebalken angezeigt. Somit wird ersichtlich, wie viel die Applikation schon

abgeschlossen hat. In dieser Zeit werden verschiedene ausgeführt. Zum einen werden die PDF's auf den FTP-Server gespeichert. Anschließend wird mit Apache Tika die Metadaten und mit PDFBox die Bilder gespeichert. Dann wird ein XML mit den verschiedenen Informationen des PDF's erstellt, welches dann vom Apache SolR indiziert wird. Zuletzt werden die Dateien, welche zuerst kopiert wurden, wieder gelöscht. Somit ist der Ordner frei für den nächsten upload. Wenn die Dateien indiziert wurden, kann die Applikation nach diesen auch suchen. Hierfür können verschiedene Kriterien angegeben werden. Zum Beispiel will der User nur nach Titeln oder Autoren suchen. Falls eine genauere Suche erwünscht wird, kann durch den Link „erw Suche" ein Fenster geöffnet werden. Hier können mehrerer Kriterien miteinander verbunden werden. Außerdem kann hier auch nach Bewertungen (Anzahl Sterne) und Zeiträumen gesucht werden. Auch die Funktion, Wörter welche nicht vorhanden sein sollen, existiert. Wenn die Suche Dokumente anzeigt, können diese durch den Link „Titelname" geöffnet werden. Außerdem kann die Indizierung durch den Link „Update" geändert werden. Das heißt, der Titel, die Autoren, die Konferenz, das Land und der Ort sind veränderbar. Falls andere Angaben falsch sein sollten, muss die Indizierung gelöscht und neu indiziert werden. Da somit die ganze Ordnerstruktur nicht mehr übereinstimmen würde. Das Löschen ist auch über den Link „Update" mit dem Button „löschen" möglich. Mit dem Link „Comment" können die Dokumente kommentiert und bewertet werden.

Dozenten	
Müller	Henning
Ivan	Eggel
Student	
Angelo	Burgener

1 Tools

Für alle möglichen Funktionen gibt es verschiedene Tools. Damit die Wahl der besten und nützlichsten Tools erleichtert wird, habe ich für jedes Gebiet eine Nutzwertanalyse von verschiedenen Programmen erstellt, welche auf der Tabelle 1 und Tabelle 2 dargestellt sind. So wird schnell ersichtlich, welches Tool meinen Erwartungen am nächsten kommt.

1.1 Mockup Tool

Tabelle 1: Nutzwertanalyse Mockup

		Balsamiq (Annen & Schmitz, 2009)			Axure (Annen & Schmitz, 2009)			Mockingbird (Pakanati & Chakrabarti, 2009)		
Kriterien	Gew.	Beschreibung	Wert	Total	Beschreibung	Wert	Total	Beschreibung	Wert	Total
Preis	7	7 Tage gratis Preis: 79 US-Dollar	1	7	30 Tage gratis Preis: 589 US-Dollar	1	7	gratis	9	63
Verfügbarkeit	5	einfacher download	9	45	einfacher download	7	35	über Web	5	25
Design	3	einfache aber saubere Darstellung der Elemente	7	21	wichtige Elemente fehlen	3	9	schlicht	3	9
Performance	5	einfache Bedienung	7	35	zu komplex	3	15	einfache Bedienung	5	25
Voraussetzungen	3	Adobe Air, wird direkt mit installiert	7	21	Lizenz nach 30 Tagen	1	3	Browser	7	21
Qualität	7	gut	7	49	gut	7	49	gut	7	49
Sprache	3	Englisch	3	9	Englisch	3	9	Englisch	3	9
Total				187			127			**201**

1.2 php Tool

Tabelle 2: Nutzwertanalyse php Tool

		php Designer (Pham, Jahr unbekannt)			oxygen (Adam, Jahr unbekannt)			PhpED (Schmitz, 2000)		
Kriterien	Gew.	Beschreibung	Wert	Total	Beschreibung	Wert	Total	Beschreibung	Wert	Total
Preis	7	gratis	7	49	gratis für uns	9	63	kostenpflichtig	3	21
Verfügbarkeit	5	einfach	5	25	einfach	5	25	einfach	5	25
Design	3	normal	3	9	gut	7	21	gut	5	15
Performance	5	normal	3	15	gut	7	35	gut	7	35
Voraussetzungen	3	keine	7	21	keine	7	21	keine	7	21
Qualität	7	normal	5	35	gut	7	49	gut	7	49
Sprache	3	englisch	5	15	deutsch/ englisch	7	21	englisch	5	15
Total				169			235			181

Leider musste ich mit der Zeit feststellen, dass dieses Tool (oxygen) nicht so grossartig lief, wie ich mir das vorgestellt habe. Die Arbeit mit diesem Tool war zu aufwändig, somit musste ich mich auf die Suche nach einer besseren und bedienbareren Umgebung begeben. Ich wurde auch ziemlich schnell fündig. Mit Netbeans, einer Umgebung die viele Optionen zulässt, wie zum Beispiel Java EE, Java FX, HTML5, … und zu meinem Vorteil PHP, wurden meine Erwartungen übertroffen. Der grösste Vorteil hierzu ist sicherlich, dass die Umgebung automatische Hilfe integriert hat, das heisst z.B. die Variablen werden im Hintergrund gespeichert und somit schnell einfügbar. So kann jede Menge Zeit und Aufwand gespart werden.

2 Services

2.1 Apache Software Foundation

2.1.1 Allgemein

Die ASF (Apache Software Foundation) ist eine gemeinnützige Gesellschaft. Von Apache Group in Delaware, USA wurde sie im Juni 1999 gegründet. Sie stützt sich auf folgende Punkte (Apache F. S., The Apache Software Foundation, Jahr unbekannt):

- bietet eine Grundlage für offene, kollaborative Software-Entwicklungsprojekte, durch eine Bereitstellung von Hardware, Kommunikation und Business-Infrastruktur
- eigenständige juristische Personen, Unternehmen und Einzelpersonen können Ressourcen spenden und sicher sein, dass diese Ressourcen für das Gemeinnutz verwendet wird
- Schutz der Marke "Apache", sowie seine Software-Produkte, damit diese von keinen anderen Organisationen missbraucht werden können

2.1.2 Geschichte

Die Apache Software Foundation erzählt ihre Geschichte folgendermassen (Apache F. S., The Apache Software Foundation, Jahr unbekannt). Die Stiftung entstand 1999 durch eine Gruppe von Menschen, die sich die "Apache Group" nannten. Sie kamen einige Jahre zuvor zusammen, um den HTTPD-Web-Server, welcher von der NCSA geschrieben wurde, weiter zu unterstützen und aufrecht zu erhalten.

Dieser Server war frei verfügbar und wurde lizensiert unter einer Lizenz, die sehr offene Modifikation und Weitergabe erlaubte. Leider verloren die ursprünglichen Entwickler das Interesse. Somit blieben alle Nutzer ohne Support.

Daher haben einige Benutzer damit begonnen, sich Fixes (so genannte „Patches“) und Informationen, wie Probleme verhindert und die Software verbessert werden kann, auszutauschen. Brian Behlendorf hat hierzu eine Mailingliste auf seiner eigenen Maschine erstellt. Diese soll die Benutzer zu einer Zusammenarbeit beheben, die Software zu pflegen und zu verbessern.

Der Name Apache wurde aus Respekt vor dem Native American Apache Nation, für ihre hervorragenden Fähigkeiten in Strategie und Kriegsführung und für ihre unerschöpfliche Ausdauer, gewählt. Hierzu gibt es auch ein nettes Wortspiel „a patchy web server“, was so viel heisst wie „ein lückenhafter Web Server“. Da dieser Server aus einer Reihe von Patches zusammengebaut wurde, ist diese Behauptung wohl nicht so falsch, allerdings wurde der Name nicht aus diesem Grund gewählt. Die Gruppe von Entwicklern, die diese neue Software veröffentlicht haben, nennen sich „Apache Group“.

Zwischen 1995 und 1999 wurde der Apache HTTPD (Hypertext Transfer Protocol Daemon) Web Server, welcher von der Apache Group entwickelt wurde, zum Marktleader. Er ist es bis heute geblieben. Mehr als 65% der Websites werden von ihm versorgt.

Als aber das Web immer grösser wurde, begann auch das wirtschaftliche Interesse zu wachsen, somit hat Apache Website neue Projekte gehostet, wie z.B. mod_ perl Perojekt, PHP Projekt, Java Apache Projekt, usw. Es wurde immer wichtiger, dass eine kohärente und strukturierte Organisation erstellt wird, welche die Menschen vor möglichen rechtlichen Angriffen abschirmen konnte.

2.2 Apache Lucene

2.2.1 Was ist Lucene?

Wie im Buch (Gospodnetic, Hatcher, & Doug, 2004, S. 7) festgehalten, wird Lucene folgendermassen beschrieben. Lucene ist eine leistungsstarke und skalierbare Informationsbibliothek, welche Indizierungen und Suchfunktionen zu ihren Anwendungen hinzufügen lässt. Sie ist ein open-source Projekt in Java implementiert und ein Mitglied der

Familie der populären Apache Jakarta, lizensiert unter der Liberalen Apache Software Lizenz. Als solches ist derzeit Lucene einer der populärsten freien Java-Bibliotheken.

Lucene bietet eine einfache, aber leistungsfähige Core –API (Application programming Interface), die minimales Verständnis der Volltextindizierung und Suche erfordert. Da sie eine Java- Bibliothek ist, wird nur eine Handvoll Klassen benötigt, um eine solche Suchfunktion in eine Anwendung einzubinden. Dies stellt einen grossen Vorteil gegenüber anderen Anbietern dar.

Menschen, die neu auf Lucene arbeiten, verwechseln sie oft mit einer ready-to-use Anwendung wie Dateisuch Programm, ein Web-Crawler oder eine Suchmaschine (Google). Das ist nicht das, was Lucene ist: Lucene ist eine Software-Bibliothek, ein Toolkit, wenn man so will, ist sie keine funktionsfähige Suchanwendung. Es speichert die Volltextindizierung und Suche selber. Lucene kann in einer Anwendung als API eingefügt werden und ist somit einfach zu bedienen.

Eine Reihe von Anwendungen, die auf Lucene aufgebaut sind, können auf der Lucene Wikiseite (Kollektiv, 2004) angesehen werden. Hier werden viele Funktionen wie z.B. zilverline, earchblox, nutch, larm, jsearch, um nur einige zu nennen angewendet.

2.2.2 Was macht Lucene?

Lucene ermöglicht es Ihnen, Indexierungen und Suchfunktionen in ihre Anwendungen hinzuzufügen. Mit ihm werden alle Daten indizier und durchsuchbar gemacht, die zu einem Textformat umgewandelt werden können, wie sie in Abbildung 2 sehen können (Gospodnetic, Hatcher, & Doug, 2004, S. 7).

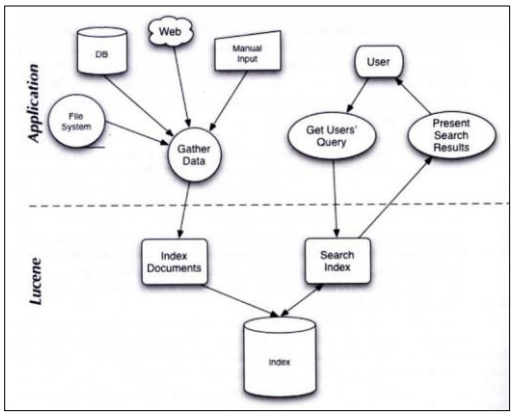

Abbildung 2: Funktionsweise Apache Lucene (Gospodnetic, Hatcher, & Doug, 2004, S. 8)

Wie im Buch (Gospodnetic, Hatcher, & Doug, 2004, S. 8) beschrieben, kümmert sich Lucene nicht um die Quellen der Daten, das Format oder sogar die Sprache, solange die Daten in Text konvertiert werden können, ist Lucene fähig, diese zu indizieren und somit die Suche zu ermöglichen. Mögliche Daten können sein: Webseiten auf Remote-Web Server, Dokumente im lokalen Dateisystem, einfache Text Dateien, Microsoft Word Dokumente, HTML Dateien, PDF Dateien oder andere Formate, welche sich in Textinformationen umwandeln lassen (siehe Abbildung 2).

2.2.3 Geschichte

Die Versionen sind von Beginn an im Jahr 2000 bis 2004 im Buch (Gospodnetic, Hatcher, & Doug, 2004, S. 9) sehr gut beschrieben. Dies wird in der nachfolgenden Tabelle ersichtlich. Leider ist das Buch vom Jahr 2004 und somit nicht vollständig mit Informationen gedeckt. Die neuste Versionen wird auf der Website (Apache F. S., Lucene, Jahr unbekannt) gut erklärt (siehe Tabelle 3).

Tabelle 3: Lucene Versionen von 2000 bis 2004 (Gospodnetic, Hatcher, & Doug, 2004, S. 9)

Version	Release Datum	Milestones
0.01	März 2000	Erstes Release (SourceForge)
1.0	Oktober 2000	
1.01b	Juli 2001	Letzter SourceForge Release
1.2	Juni 2002	Erste Apache Jakarta Release
1.3	Dezember 2003	compound index format, queryparser enhancements, remote searching, token positioning ,extensible scoring api
1.4	Juli 2004	Sortieren, span queries, term vectors
1.4.1	August 2004	Korrigierter Fehler beim der sortierperformance
1.4.2	Oktober 2004	IndexSearcher optimiert
1.4.3	Winter 2004	verschiedene Korrekturen

2.2.4 Wie funktioniert indexing ?

Nach dem Buch (Gospodnetic, Hatcher, & Doug, 2004, S. 29) wird das Indexieren folgendermassen erklärt. Jedes Format, das verwendet werden soll, muss zuerst in ein Textformat umgewandelt werden. Lucene kann nur mit diesen Textformaten arbeiten. Wenn

direkt eine Textdatei verwendet wird, ist die Arbeit mit Lucene relativ einfach. Allerdings bei anderen Formaten muss ein kleiner Umweg vorgenommen werden.

Angenommen, eine Reihe von Handbüchern im PDF Format muss indiziert werden. Damit Lucene mit diesen Dateien arbeiten kann, muss ein Weg gefunden werden, um die Textinformationen aus den PDF Dokumenten extrahieren zu können und diese dann in ein Dokument zu integrieren, welche aus den Feldern besteht, mit denen Lucene arbeiten kann. Das gleiche gilt auch für die anderen Formate wie z.B. Microsoft Word Dokumenten. Auch bei HTML oder XML Dateien, welche schon Klartext Zeichen verwenden, muss ein neues Dokument erstellt werden, welches die erforderlichen Tags aufweisen.

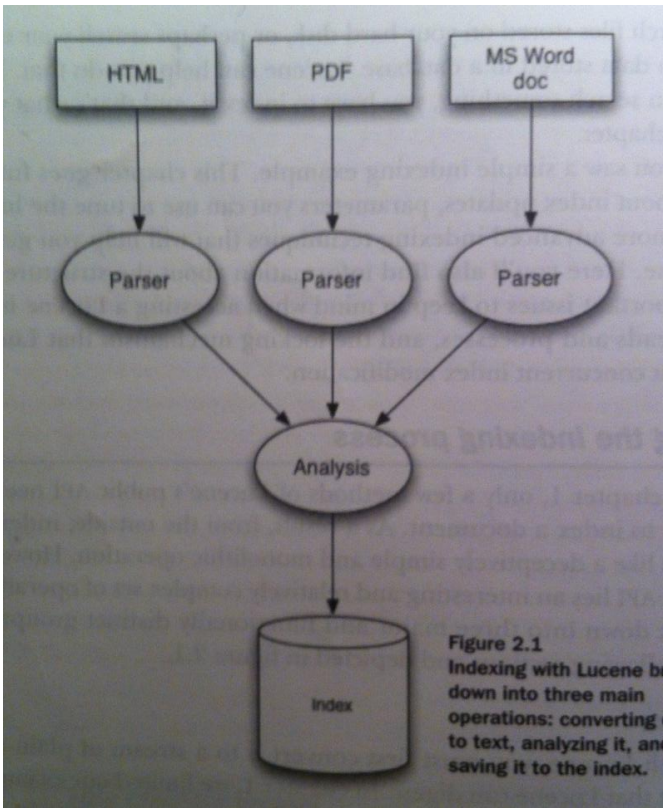

Abbildung 3: Funktionsweise Indizierung (Gospodnetic, Hatcher, & Doug, 2004, S. 30)

Nachdem die Daten für die Indizierung vorbereitet und ein Lucene-Dokument erstellt wurde, kann z.B. in einem Java Projekt der Indexwriter mit der Methode adddocument(document) aufgerufen werden. Wenn dies getan ist, analysiert Lucene zuerst die Daten, ob diese auch für das indizieren geeignet sind. Hierzu werden die Textdaten in einzelne Brocken oder Token geteilt. Auf diesen Einzelteilen werden anschliessend verschiedene optionale Operationen durchgeführt. Zum Beispiel kann die Suche bei Gross-

und Kleinschreibung vermieden werden, indem eine dieser Operationen alles in Kleinbuchstaben speichert. Dieser Schritt nennt man Analyse, wie in Abbildung 3 ersichtlich. Die Daten können in verschiedener Art und Weise analysiert werden. Dieser Schritt kann auch als eine Art Filter betrachtet werden (siehe Abbildung 3).

Nachdem die Eingabe analysiert wurde, ist sie bereit in den Index aufgenommen zu werden. Lucene speichert die Eingabe in eine Datenstruktur. Es werden nur die einzelnen Token gespeichert. Die Suche erfolgt dann nicht über das Original Dokument sondern ausschliesslich über die einzelnen Token. Mit anderen Worten, anstatt zu versuchen die Antwort auf die Frage „Welche Wörter sind in diesen Dokumenten enthalten?“ zu finden, wird die Antwort für die Frage „Welche Dokumente enthalten das Wort X?“ ausgegeben. Wenn dies mit einer normalen Websuchmaschine verglichen wird, wird festgestellt, dass genau die zweite Frage so schnell wie möglich beantwortet werden soll. Der Kern aller heutigen Websuchmaschinen besteht aus Indizes. Jede Suchmaschine unterscheidet sich in den verschiedenen Parametern, welche zusätzlich hinzugefügt werden können.

2.3 Apache SolR

Auf der Seite (Apache S. F., Apache Solr, 2011) wird Apache SolR folgendermassen erklärt. SolR ist eine beliebte und extrem schnelle Open-Source-Enterprise-Search-Plattform von Apache Lucene. Seine wichtigsten Merkmale sind leistungsstarke Volltextsuche, hit highlighting, facettierte Suche, Indizierung in Echtzeit, dynamische Clustering, Datenbank-Integration und Handhabung von verschiedenen Dokumenten (z.B. Word, PDF). SolR ist höchst zuverlässig, skalierbar und fehlertolerant, bietet verteilte Indizierung an, hat eine zentrale Konfiguration und vieles mehr. SolR wird von vielen der weltweit grössten Internetseiten verwendet.

SolR ist in Java geschrieben und läuft als alleinstehender Volltextsuchserver innerhalb eines Servlet-Container wie Jetty. Es verwendet in seinem Kern die Lucene Java Suchbibliothek für die Volltextindizierung und Suche. Ausserdem hat SolR REST wie HTTP/XML und JSON, was es relativ einfach macht mit nahezu jeder Programmiersprache damit zu arbeiten. SolR’s externe Konfiguration ermöglicht die Zusammenarbeit mit nahezu jeder Art von Anwendungen ohne eine Java-Codierung. Zusätzlich verfügt es über eine umfangreiche Plugin-Architektur, falls erweiterte Anpassungen erforderlich sind.

- erweiterte Volltext- Suchfunktionen
- optimiert für High Volume Web Traffic
- standartisierte basis open Interfaces – XML, JSON und http
- umfassende HTML –Administrations Interfaces
- Serverstatistiken über JMX zur Überwachung
- linear skalierbar
- real-time indexing
- flexibel
- erweiterbare Plugin Architektur

2.4 Apache Tika

Um mit PDF's arbeiten zu können, muss der Zugriff auf den Inhalt und die Metadaten der Datei ermöglicht werden. Hierzu wird der Service Apache Tika verwendet. Dieser Service ist, wie auch Apache SolR, in Java geschrieben. Damit ich es in PHP verwenden kann muss ich eine library (jar-File) mit der „exec()" Funktion aufrufen. Dabei stehen verschiedene Funktionen zur Verfügung, wie in Abbildung 4: Apache Tika Funktionen ersichtlich ist.

Für meine Arbeit habe ich zwei Funktionen verwendet. Zuerst „-m" für die Metadaten. Hier erhalte ich den Filename, Anzahl Seiten und das Erstellungsdatum. Dann die Funktion „-t" für den Inhalt. Mit dem Inhalt erstellte ich den Titel, Autor und die ersten Zeilen des Inhalts.

```
Options:
    -?  or --help          Print this usage message
    -v  or --verbose       Print debug level messages
    -V  or --version       Print the Apache Tika version nu

    -g  or --gui           Start the Apache Tika GUI
    -s  or --server        Start the Apache Tika server
    -f  or --fork          Use Fork Mode for out-of-process

    -x  or --xml           Output XHTML content (default)
    -h  or --html          Output HTML content
    -t  or --text          Output plain text content
    -T  or --text-main     Output plain text content (main
    -m  or --metadata      Output only metadata
    -j  or --json          Output metadata in JSON
    -y  or --xmp           Output metadata in XMP
    -l  or --language      Output only language
    -d  or --detect        Detect document type
    -eX or --encoding=X    Use output encoding X
    -pX or --password=X    Use document password X
    -z  or --extract       Extract all attachements into cu
    --extract-dir=<dir>    Specify target directory for -z
    -r  or --pretty-print  For XML and XHTML outputs, adds
                           whitespace, for better readabili
```

Abbildung 4: Apache Tika Funktionen (Apache S. F., Apache Tika, 2013)

2.5 PDFBox

Leider war es nicht möglich mit Apache Tika auch Bilder zu extrahieren. Somit musste ich einen zusätzlichen Service einbauen mit dem Namen PDFBox. Mit seiner Hilfe wird dies ermöglicht. Auch dieser Service ist in Java geschrieben und muss über ein jar-File aufgerufen werden.

2.6 Server

2.6.1 wamp

Wie der Autor (Bourdon, Jahr unbekannt) schreibt, handelt es sich bei WAMP um eine Windows Web Entwicklungsumgebung. Mit diesem Server wird eine Erstellung von Webapplikationen mit Apache2, PHP und einer MySQL Datenbank ermöglicht. Zusätzlich kann mit Hilfe von PhpMyAdmin eine einfache Verwaltung der Datenbank gewährleistet werden.

Der Wamp Server wird dazu verwendet um eine Applikation auch lokal zu testen und anzeigen zu lassen. Für die Entwicklung ist dies sicherlich eine einfache und schnelle Variante um z.B. eine WebApplikation ausserhalb eines Servers zu entwickeln.

2.6.2 medgift.hevs.ch

Damit die Applikation nicht nur lokal funktioniert und angewendet werden kann, wird ein Server (medgift.hevs.ch) zur Verfügung gestellt. So kann die Applikation auch von extern genutzt werden. Der Server läuft auf einem Linux Betriebssystem (siehe Abbildung 5).

Abbildung 5: Server medgift.hevs.ch Informationen

3 Mockups

3.1 Web

Für die gedankliche Vorstellung der Webapplikation wurden Mockups für jede Seite erstellt. In Abbildung 6 findet sich das Mockup für das Login.

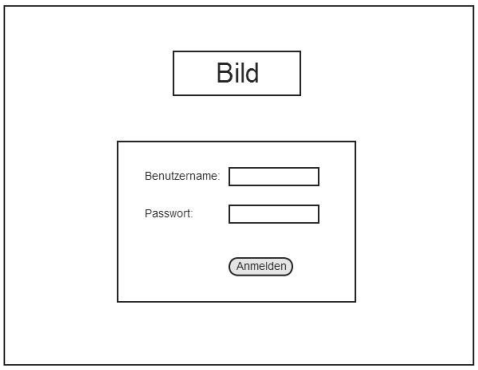

Abbildung 6: GUI Login

In Abbildung 7 findet sich das Mockup für die normale Darstellung (Main).

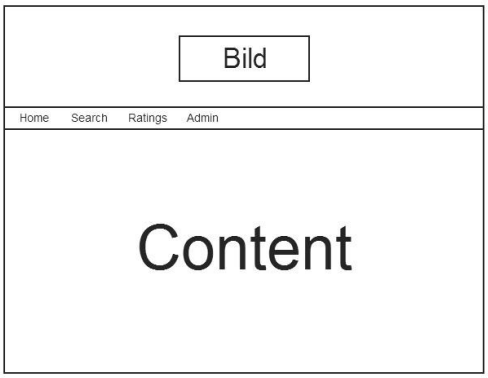

Abbildung 7: GUI Main

In Abbildung 8 findet sich das Mockup für die Startseite.

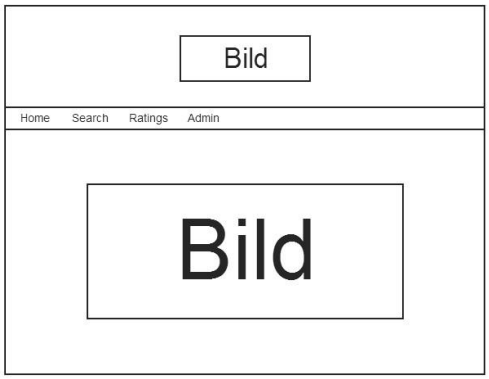

Abbildung 8: GUI Home

In Abbildung 9 findet sich das Mockup für die Suche.

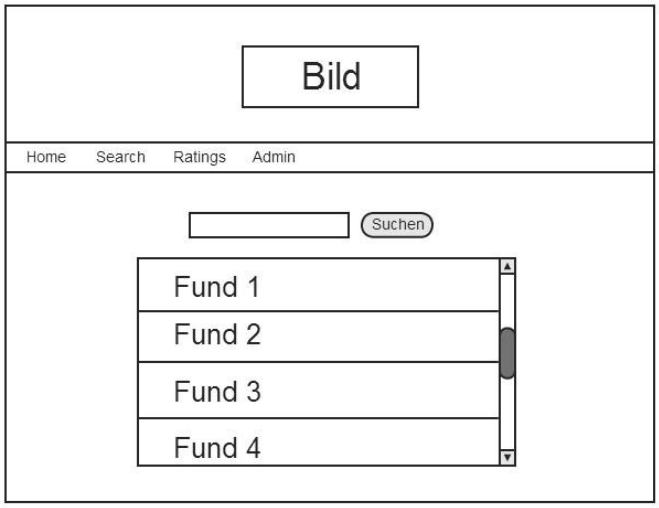

Abbildung 9: GUI Search

Im Register Admin hat man die Möglichkeit sich abzumelden oder ein neues Profil zu erstellen. Dies ist erst möglich, wenn der eine Admin eingeloggt ist, ansonsten kann kein neuer User erstellt werden. Außerdem kann der User im Register Admin sein Passwort ändern.

3.2 Mobil

Auch für das Mobile Interface wurden Mockups erstellt. In Abbildung 10 findet sich das Login, in Abbildung 12 findet sich die Suche und in Abbildung 11 findet sich die erweiterte Suche.

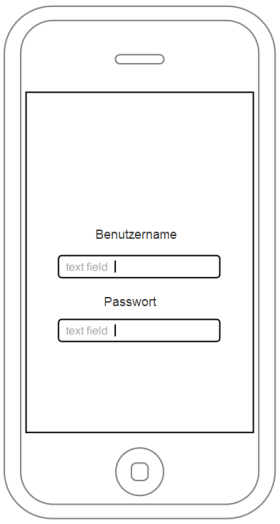

Abbildung 10: GUI Mobile Login

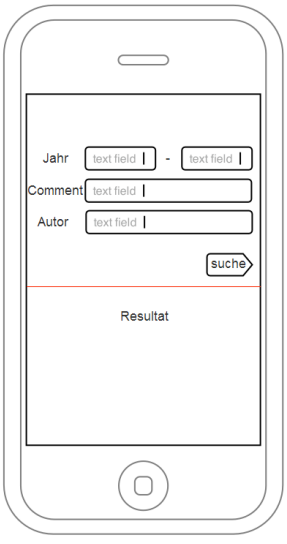

Abbildung 11: GUI Mobile erweiterte Suche

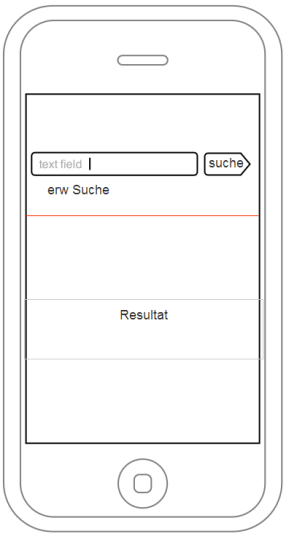

Abbildung 12: GUI Mobile Suche

4 Code

4.1 Login

4.1.1 GUI

Damit nicht jeder von der Anwendung Gebrauch nehmen kann, wird ein Login eingebaut. Dies stellt sicher, dass nur Befugte zur Anwendung zugreifen können. Damit ein Login erstellt werden kann, braucht es ein GUI. Hier werden die erforderlichen Felder bereitgestellt, damit der User sich über ein Benutzername und ein Passwort einloggen kann.

Abbildung 13: GUI Login

Um die Darstellung genauer zu definieren werden zusätzlich CSS Dokumente erstellt. Hier wird genau definiert, wie die Seite dargestellt werden soll. Für jede spezifische Seite wird ein neues Dokument erstellt. So kann die Übersichtlichkeit gewährleistet werden. Damit die Anwendung immer das gleiche Layout aufweist, werden spezielle CSS Dokumente erstellt.

- style.css
- style_aenderungen.css
- style_navigation.css

Diese werden dann auf jeder Seite angewendet, so bleibt die Struktur der Seiten immer gleich. Die Grundstruktur des Layouts und somit der CSS Files (style.css und style_navigation.css) sind aus einem Standartlayout von Microsoft Visual Studio entnommen. Alle Anpassungen sind im “style_aenderungen.css“ enthalten.

Um das Login zu definieren wird ein Dokument mit dem Namen „Login.css“ erstellt. So ist schnell ersichtlich für welche Seite das CSS benötigt wird. Die Seite wird anschließend, wie in Abbildung 13, dargestellt.

4.1.2 Datenbank

Jeder User, der Zugriff auf die Anwendung hat, besitzt ein Benutzername und ein Passwort. Um diese zu speichern wird eine Datenbank (MySQL) verwendet. Hier werden genau diese Informationen gelagert und bei Bedarf verwendet.

Die Tabelle, die verwendet wird, speichert nicht nur den Benutzername und Passwort, sondern auch der Name, Vorname und E-Mail der Person. Die Struktur sieht wie folgt aus (siehe Abbildung 14):

	#	Name	Typ	Kollation	Attribute	Null	Standard	Extra
☐	1	ID	int(11)			Nein	*kein(e)*	AUTO_INCREMENT
☐	2	user	varchar(20)	latin1_swedish_ci		Nein	*kein(e)*	
☐	3	pwd	varchar(20)	latin1_swedish_ci		Nein	*kein(e)*	
☐	4	name	varchar(20)	latin1_swedish_ci		Nein	*kein(e)*	
☐	5	vorname	varchar(20)	latin1_swedish_ci		Nein	*kein(e)*	
☐	6	email	varchar(50)	latin1_swedish_ci		Nein	*kein(e)*	

Abbildung 14: Datenbank Struktur Tabelle users

Und so könnte ein Datensatz aussehen (siehe Abbildung 15):

		ID	user	pwd	name	vorname	email
☐	Bearbeiten Kopieren Löschen	1	test	welcome	muster	hans	hans.muster@hotmail.com

Abbildung 15: Datensatz Tabelle users

Mit diesen Informationen ist es jetzt möglich, ein Login durchzuführen. Damit nicht jeder einfach ein Login erstellen kann, wird ein Hinzufügen eines Benutzers erst nach einem erfolgreichen Einloggen ermöglicht. So können unerwünschte Personen von der Anwendung ferngehalten werden.

4.1.3 Klassen

Diese zwei Elemente, GUI und Datenbank, müssen nun miteinander kommunizieren können. Hierzu werden PHP Klassen erstellt. Zuerst muss eine Verbindung zur Datenbank erstellt werden. Dies erfolgt mit der Klasse „dbConnection.php“ (siehe Abbildung 16).

```
class DB {
    // Felder
    /* Connect to an ODBC database using driver invocation */
    private static $dsn = 'mysql:dbname=digital_library;host=127.0.0.1';
    private static $user = 'admin';
    private static $password = 'welcome';
    static $dbh;


    //Stellt die Verbindung zur Datenbank her
    static function connect() {
        //__construct
        //achtung hier klammern bei umkehrung von negation

        if(!(DB::$dbh instanceof PDO)){
            try {
                //ohne this da static not insantzierung
                //DB::$dbh =new PDO($this->dsn, $this->user, $this->password);
                DB::$dbh =new PDO(DB::$dsn, DB::$user, DB::$password);
                }
                catch (PDOException $e) {
                echo 'Connection failed: ' . $e->getMessage();
                }
        }
    }

    //gibt die Datenbankverbindung zur&uuml;ck
    public static function getDBH(){
       DB::connect();
       return DB::$dbh;
       }

}
```

Abbildung 16: Datenbankverbindung dbConnetion.php

In dieser Klasse werden die Anmeldeinformationen wie Datenbankname, User und Passwort gespeichert. Mit diesen Daten kann dann eine Verbindung erstellt werden. Jenes geschieht durch einen Aufruf der Methode „getDBH()“, welche dann die „connect()“ Methode aufruft. Hier wird die Verbindung erstellt und zurückgegeben. So erhält das GUI eine Verbindung zur Datenbank.

Jetzt müssen allerdings die Informationen aus der Datenbank geholt werden, damit auch kontrolliert werden kann, ob die eingegebenen Angaben auch korrekt sind. Diese Aufgabe übernimmt eine andere Klasse „UserdatenLesen.php“ (siehe Abbildung 17).

```
include_once 'dbConnection.php';

class Benutzerdaten extends DB {

        //Array f□r die Benutzerdaten
        static $liste = array();


        //Funktion, um die Benutzerdaten in das oben angelegte Array zu speichern
        static function beutzerdatenAuslesen(){
                $sql = 'SELECT ID, user, pwd, name, vorname, email from users';
                $array = parent::getdbh()->query($sql);
                foreach ($array as $row) {
                Benutzerdaten::$liste[]=new Benutzerdaten($row['ID'], $row['user'],
        }

}
```

Abbildung 17: Datenbank lesen UserdatenLesen.php

Hier wird ein Array erstellt, welches eine Instanz pro Datensatz speichert. Diese Instanz enthält alle Informationen, die in der Datenbank gespeichert wurden. So wird es auch möglich die eingegebenen Daten mit den gespeicherten zu vergleichen. Falls ein Datensatz mit den Angaben übereinstimmen, wird der User auf die Hauptseite verlinkt.

```
//Benutzernamen und Passwort aus Liste vergleichen
foreach (Benutzerdaten::$liste as $daten){
         if(strcmp($daten->getUser(), $benutzer)==0){
                 if(strcmp($daten->getPwd(), $pw)==0){
                 //Session anlegen f□r den gew□hlten Benutzer
                 //ID des Benutzer in Sessionvariable speichern
                 if (!isset($_SESSION['benutzer'])){
                         $benutzerID=$daten->getId();
                         $_SESSION['benutzer'] = $benutzerID;
                 }
                 //Verlinkung zur Hauptseite
                 header ("Location: GUIMain.php");
                 break;
                 }
          }
 }
```

Abbildung 18: Loginkontrolle GUILogin.php

Zusätzlich wird eine Session erstellt. Diese hat mehrere Nutzen. Zum Einen kann die BenutzerID gespeichert werden (siehe Abbildung 18). Somit ist auf jeder Seite bekannt, welcher User gerade an der Anwendung arbeitet. Zum Anderen dient sie zur Kontrolle. Die

Möglichkeit soll nicht bestehen, dass ein User, durch Eingabe einer URL, das Login umgehen kann. Somit wird bei jedem GUI Dokument zuerst die Session kontrolliert, dies erfolgt durch die Klasse „SessionKontrolle.php“ (siehe Abbildung 19).

```php
<?php session_start();
if ($_SESSION['benutzer']=="") {
        header("Location: GUILogin.php");
}
?>
```

Abbildung 19: Sessionkontrolle SessionKontrolle.php

In dieser Klasse wird kontrolliert, ob eine Session schon existiert. Falls dies nicht der Fall sein sollte, wird der User direkt zum Login verlinkt. Somit kann das Login nicht mehr umgangen werden.

4.2 Logout

4.2.1 GUI

Wenn der User sich ausloggt, wird er auf das File „Logout.php“ verlinkt. Hier wird ihm, durch eine Ausgabe eines Satzes „Sie haben sich erfolgreich abgemeldet“, bestätigt, dass er ausgeloggt ist (siehe Abbildung 20).

```php
<form action="GUILogin.php" method="post">
<fieldset>
<p>
<label class="logoutText">
Sie haben sich erfolgreich abgemeldet
</label>
</p>
</fieldset>
<p>
        <input type="submit" name="logout" value="zum Login" class="logoutButton"/>
</p>
</form>
```

Abbildung 20: Logout Logout.php

Außerdem hat er hier die Möglichkeit sich wieder auf die Login Seite zu verlinken. Dies wird durch das Betätigen des Buttons „zum Login“ ermöglicht.

4.3 Main

4.3.1 GUI

Dieses File „GUIMain.php“ wird nicht für die Anwendung verwendet. Sie dient ausschließlich der Entwicklung der Anwendung bei. In ihr ist die Grundstruktur gespeichert, welche in jedem GUI-File, außer im „GUILogin.php“, enthalten sein muss.

```
<?php
include_once("klassen/SessionKontrolle.php");
include 'klassen/mainDir.php';
?>

<!DOCTYPE HTML>
<head>
<title>Digital libary</title>
<?php include("Head.php"); ?>
</head>

<body>
        <div class="page">
                <!-- Container f&uuml;r Inhalte -->
                <!-- Eif&uuml;gen Header und Navigation -->
                <?php include("Navigation.php"); ?>

                <!-- Ab hier beginnt die eigentliche seite -->

                <div class="main">

                        <div id="content">
                                <section id="content">
                                        <article title="">

                                        </article>
                                </section>
                        </div>
                </div>
   <div class="clear">
</div>
        </div>
        <!-- Ende der klasse page -->
```

Abbildung 21: Grundstruktur GUIMain.php

Die Struktur sieht so aus, wie in Abbildung 21 ersichtlich ist. Zuerst wird die Session kontrolliert, durch das Einbinden der Klasse „SessionKontrolle.php“. Anschließend wird der Header definiert. Hier wird der Titel angegeben und alle Verlinkungen zu CSS und JS

eingebunden, durch das File „Head.php“. Dann wird der Body mit verschiedenen Div-Tags strukturiert. Auch die Navigation wird durch ein Include von „Navigation.php“ eingebunden.

Diese Elemente bleiben bei jedem GUI gleich. Die spezifischen Elemente werden im Tag <article> eingebunden.

4.3.2 Navigation

Das File „Navigation.php“ enthält alle Register. Jedes Register kann mehrere Unterregister enthalten.

```
<div class="clear hideSkiplink">
        <!-- Navigation -->
        <nav>
                <ul id="sddm">
                        <li><a href="GUIHome.php" onmouseover="mopen('m1')" onmouseout="mclosetime()">Home</a></li>
                        <li><a href="GUISearch.php" onmouseover="mopen('m2')" onmouseout="mclosetime()">Search</a></li>
```

Abbildung 22: Navigation Register Navigation.php

Durch ein Hinzufügen von li-Tags können weitere Register erstellt werden (siehe Abbildung 22).

```
<li><a href="#" onmouseover="mopen('m10')" onmouseout="mclosetime()">Admin</a>
        <div id="m10" onmouseover="mcancelclosetime()"
                onmouseout="mclosetime()">

                <a href="GUIAddPDF.php">PDF hinzuf&uuml;gen</a>
                <a href="GUILogout.php">Logout</a>
        </div></li>
```

Abbildung 23: Navigation Unterregister Navigation.php

Wie in Abbildung 23 ersichtlich, können Unterregister einfach mit einem Link hinzugefügt werden. In diesem Beispiel wird im Register „Admin“ zwei Unterregister mit den Namen „PDF hinzufügen“ und „Logout“ eingebunden.

```
<!-- Suchfunktion in Navigation einbinden -->
<li>
    <form action="GUISearch.php" method="post">
        <select class="suchfunktion" name="art" size="1">
                                    <option>*</option>
                                    <option>filename</option>
                                    <option>title</option>
                                    <option>location</option>
                                    <option>conference</option>
                                    <option>author</option>
                                    <option>year</option>
                                </select>
        <input type="text" name="suche">
        <input type="submit" value="suchen" name="sucheBnt">
    </form>
</li>
```

Abbildung 24: Navigation Suchfunktion Navigation.php

Damit die Suchfunktion immer zur Verfügung steht, wird sie in der Navigation integriert. Somit hat der User die Möglichkeit, in jeder Situation nach PDF's zu suchen. Um die Suche genauer zu definieren ist eine Dropdownliste mit den möglichen Suchfeldern eingebunden. Bei einer Eingabe von * werden in allen Felder nach dem bestimmten Begriff gesucht (siehe Abbildung 24).

4.4 Home

4.4.1 GUI

Die Seite Home „GUIHome.php“ enthält keine Funktionen. Sie dient als Startseite für die Anwendung nach dem Login. Auf der Anzeige wird ein Bild der digital library angezeigt. Somit wird ersichtlich, dass sich der User auf dem Home Register befindet. Für die richtige Darstellung ist das CSS File „Home.css“ erstellt worden. Hier wird das Homebild exakt positioniert.

4.5 AddPDF

4.5.1 GUI

Um ein PDF richtig zu speichern und zu indizieren, müssen verschiedene Eingaben manuell betätigt werden.

```
<form action="" method="post">
    <table>
        <tr>
            <td>
                <div class="addContent">
                    <table>
                        <tr>
                            <td>Jahr:</td>
                            <td><input type="text" name="jahr"></td>
                        </tr>
                        <tr>
                            <td>Kurzname Konf/Jou:</td>
                            <td><input type="text" name="confkurz"></td>
                        </tr>
                        <tr>
                            <td>Konferenz/Journal:</td>
                            <td><input type="text" name="conf"></td>
                        </tr>
                        <tr>
                            <td>Ort:</td>
                            <td><input type="text" name="ort"></td>
                        </tr>
                        <tr>
                            <td></td>
                            <td><input type="submit" value="    ok    " name="ok"></td>
                        </tr>
                    </table>
                </div>
            </td>
```

Abbildung 25: Addfunktion HTML Teil GUIAddPDF.php

Zuerst muss das Jahr und die Abkürzung für die Konferenz oder Journal eingegeben werden. Diese Angaben werden zu Erstellung der Ordnerstruktur verwendet. Weitere Infos, wie der ausgeschriebene Name der Konferenz / Journal oder das Land/ der Ort, werden zusätzlich für alle PDF‘ s, die zusammen gespeichert werden, indiziert (siehe Abbildung 25). Somit hat der User die Möglichkeit auch nach diesen Kriterien zu suchen. Damit dies auch funktionieren kann, braucht es einen PHP – Teil. Hier werden alle Apache Elemente eingebunden:

- Apache SolR
- Apache Tika

```php
<td>
    <div class="ladeContent">
        <?php
            If (isset ($_POST['ok']))  {
                //Werte in variablen speichern
                $jahr = $_POST['jahr'];
                $confkurz = $_POST['confkurz'];
                $ort = $_POST['ort'];
                $conf = $_POST['conf'];

                //Die Namen aller abgelegten Dateien werden in ein Array gespeichert
                if ($verzeichnis = opendir(  $mainDir . '/Health/upload'  )) {
                    $dateinamen = array();

                    while (false !== ($datein = readdir($verzeichnis))) {
                        if ($datein!="." && $datein!="..") {
                            $dateinamen[] = $datein;
                        }
                    }

                    closedir($verzeichnis);
                }
```

Abbildung 26: Addfunktion PHP Teil 1 GUIAddPDF.php

Für das Speichern der PDF's wird ein bestimmter Ordner (Health/upload) verwendet (siehe Abbildung 28). Alle darin enthaltenen PDF Dateien werden in die Library gespeichert und indiziert (siehe Abbildung 27). Damit das System weiß, um welche Dateien es sich handelt, werden bei der Abbildung 26 in der „while" Schleife alle Dateinamen in ein Array gespeichert.

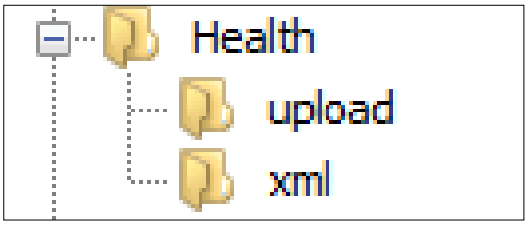

Abbildung 28: Ordnerstruktur Health

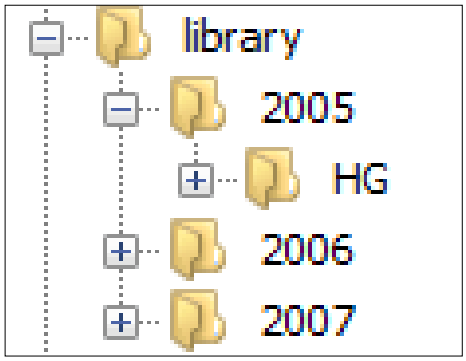

Abbildung 27: Ordnerstruktur library

```php
//Ladebalken 1 anlegen
include 'klassen/progressbar.php';
$myprogressbar1 = new progressbar(0, count($dateinamen), 300, 20, '#00f');
$myprogressbar1->set_show_digits(false);
$myprogressbar1->print_code();

//Error handler für mkdir()
function err_handler($errno, $errstr) {
    // Ignorieren
}
//handler wird gesetzt
set_error_handler('err_handler');

// Ziel Verzeichnis festlegen
$destinationDir1 = $mainDir . '/library/' . $jahr;
mkdir($destinationDir1);
$destinationDir2 = $mainDir . '/library/' . $jahr . '/' . $confkurz;
mkdir($destinationDir2);
```

Abbildung 29: Addfunktion PHP Teil 2 GUIAddPDF.php

Jede Speicherung und Indizierung braucht seine Zeit. Viele Funktionen werden für jede Datei ausgeführt. Somit ist es interessant zu sehen, wo sich das System aktuell befindet. Hierzu wurde ein Ladebalken eingefügt (siehe Abbildung 29).

Der Errorhandler dient dazu bei, dass kein Fehler ausgegeben wird, falls die Ordnerstruktur, die erstellt werden möchte, bereits vorhanden ist.

Mit den Angaben Jahr und der Abkürzung der Konferenz/Journal wird die Ordnerstruktur mit „mkdir“ erstellt. Falls ein Ordner schon vorhanden sein sollte, wird der Errorhandler aufgerufen. So kann dieser Fehler abgefangen werden.

```php
//Jede Datei wird auf den FTP-Server gespeichert
//Für jede Datei werden die Metadaten gespeichert
//Für jede Datei wird ein XML erstellt
foreach ($dateinamen as $dateiname) {
    // sourcedatei definieren
    $sourceDir = $mainDir . '/Health/upload/';
    $file = $sourceDir . $dateiname;

    //Ziel definieren
    $remote_file = $destinationDir2 . '/' . $dateiname;

    // Datei hochladen
    if (copy($file, $remote_file)) {
     echo "$dateiname erfolgreich hochgeladen<br />";
    } else {
     echo "Ein Fehler trat beim Hochladen von $file<br />";
    }
```

Abbildung 30: Addfunktion PHP Teil 3 GUIAddPDF.php

Mit den vorhin gespeicherten Namen der Dateien, wird nun eine Schleife abgelaufen, welche für jede PDF – Datei die nötigen Funktionen ausführt (siehe Abbildung 30). Zuerst wird der Dateiname mit dem Pfad gespeichert. „$mainDir“ ist eine globale Variable. Sie wird in der Datei „mainDir.php“ definiert. Hier wird der Pfad gespeichert, wo sich das Projekt befindet. So muss, bei einer Verschiebung des Projekts, nur diese Datei angepasst werden.

Als Ziel wird die zuvor erstellte Ordnerstruktur verwendet mit dem aktuellen Namen. Wenn Source- und Zielpfad definiert sind, kann die Datei mit „copy()“ kopiert werden.

```
                            //Klasse Tika aufrufen
                            include_once("klassen/Tika.php");
                            $tika = new Tika($dateiname);

                            //XML erstellen
                            include_once("klassen/XML.php");
                            new XML($tika, $conf, $ort, $jahr, $confkurz);

                            //je Schlaufe einen Schritt weiter
                            $myprogressbar1->step();
                        }

                        //Ladebalken beendet
                        $myprogressbar1->complete();

                        //XML auf SolR indizieren
                        include_once("klassen/SolR.php");
                        $solr = new Solr();
                        $solr->post();

                        //zum upload bereitgelegte PDF-Dateien wieder löschen
                        foreach (glob($mainDir . "/Health/upload/*.pdf") as $filename) {
                            unlink($filename);
                        }
                    }
                ?>
            </div>
        </td>
    </tr>
</table>
</form>
```

Abbildung 31: Addfunktion PHP Teil 4 GUIAddPDF.php

Anschließend werden nun die Klassen „Tika“ und „XML“ aufgerufen (siehe Abbildung 31). Diese werden im nächsten Unterkapitel 4.5.2 genauer beschrieben.

Damit der Ladebalken auch eine Funktion erhält, wird die Methode „step()“ pro Schleifendurchgang aufgerufen. So wird nach jeder Datei der Ladebalken einen Fortschritt machen. Nach Vollendung der Schleife wird der Ladebalken mit der Methode „complete()“ beendet.

Dann werden alle XML – Dateien, welche in der Klasse „XML“ erstellt werden, indiziert. Dies erfolgt durch die Klasse „Solr“ mit der Methode „post()“. Zum Schluss werden alle PDF’s, welche in den Ordner upload kopiert wurden, gelöscht.

Auch hier wird für die Darstellung ein CSS File mit dem Namen „AddPDF.css“ verwendet. Alle Positionierungen werden hier vorgenommen.

4.5.2 Klassen

Für das GUI „GUIAddPDF.php“ werden viele Klassen verwendet. Hier ist die größte Arbeit für das System enthalten.

```php
<?php
    $mainDir = 'c:/wamp/www';
?>
```

Abbildung 32: Mainpfad Projekt mainDir.php

In der Datei „mainDir.php“ wird der Pfad zum Projekt angegeben (siehe Abbildung 32). Somit kann, bei einer Verschiebung des Projekts, ganz einfach die Anpassungen vorgenommen werden.

```php
class Tika
{

    public $filename;
    public $creatDate;
    public $pages;
    public $title;
    public $author;
    public $content = array();


    function __construct($file) {
        include 'mainDir.php';
        $metadata = shell_exec('java -jar lib/tika-app-1.3.jar -eUTF-8 -m '.$mainDir.'/Health/upload/' . $file);
        $cont = shell_exec('java -jar lib/tika-app-1.3.jar -eUTF-8 -t '.$mainDir.'/Health/upload/' . $file);

        //Alle wichtigen Elemente ausfiltern
        //Filename
        $filenameTmp1 = explode('resourceName: ', $metadata);
        $filenameTmp2 = explode('.pdf', $filenameTmp1[1]);
        $this->filename = $filenameTmp2[0] . '.pdf';

        //Erstellungsdatum
        $creatDateTmp1 = explode('Creation-Date: ', $metadata);
        $creatDateTmp2 = explode('Z', $creatDateTmp1[1]);
        $this->creatDate = $creatDateTmp2[0] . 'Z';

        //Anzahl Seiten
        $pagesTmp1 = explode('xmpTPg:NPages: ', $metadata);
        $tmpPages = substr($pagesTmp1[1], 0, -1);
        $this->pages = $tmpPages;
```

Abbildung 33: Klasse Tika Tika.php

Damit mit PDF's gearbeitet werden kann, müssen sie als TXT vorhanden sein. Dies ermöglicht uns Apache Tika (siehe Abbildung 33). Zum Einen können wir die Metadaten der Datei erhalten, welche in die Variable „$metadata" gespeichert werden und zum Anderen kann der ganze Inhalt als Text gespeichert werden. Dies erfolgt mit der Variable „$cont". Mit diesen zwei Variablen können nun die einzelnen gewünschten Kriterien herausgeholt oder geschnitten werden.

```
public function getFilename() {
    return $this->filename;
}

public function getDate() {
    return $this->creatDate;
}
```

Abbildung 34: Klasse Tika getMethoden Tika.php

Um anschließend aus anderen Klassen auf diese Informationen zugreifen zu können, werden get - Methoden eingefügt (siehe Abbildung 34).

```
class XML
{

    function __construct($tika, $conf, $ort, $jahr, $confKurz) {
        include 'mainDir.php';
        $content = $tika->getContent();

        $xml = "<add>\n";
        $xml .= "<doc>\n";

        $xml .= "<field name=\"id\">".$tika->getFilename() . $tika->getPages()."</field>\n";
        $xml .= "<field name=\"filename\">".$tika->getFilename()."</field>\n";
        $xml .= "<field name=\"date\">".$tika->getDate()."</field>\n";
        $xml .= "<field name=\"pages\">".$tika->getPages()."</field>\n";
        $xml .= "<field name=\"title\">".$tika->getTitle()."</field>\n";
        $xml .= "<field name=\"author\">".$tika->getAuthor()."</field>\n";
        $xml .= "<field name=\"year\">".$jahr."</field>\n";
        $xml .= "<field name=\"conference\">".$conf."</field>\n";
        $xml .= "<field name=\"location\">".$ort."</field>\n";
        $xml .= "<field name=\"conf\">".$confKurz."</field>\n";
        $xml .= "<field name=\"content\">".$content[0]."</field>\n";
```

Abbildung 35: Klasse XML Teil 1 XML.php

```
    $xml .= "</doc>\n";
    $xml .= "</add>";

    $xml = utf8_encode($xml);


    $element = new SimpleXMLElement($xml);
    $name = substr($tika->getFilename(), 0, -4);
    $element->saveXML($mainDir . "/Health/xml/" . $name . ".xml");

}
```

Abbildung 36: Klasse XML Teil 2 XML.php

Bis jetzt wurden erst die Daten aus dem PDF herausgeholt und gespeichert. Jetzt müssen diese Dateien für das indizieren vorbereitet werden. Der Service Apache SolR arbeitet ausschließlich mit XML – Dateien. Solche werden mit Hilfe der Klasse „XML“ erstellt (siehe Abbildung 35 und Abbildung 36). Hier werden die Angaben vom User, wie Jahr oder Ort, verwendet und eine Instanz zur Klasse „Tika“ als Parameter übergeben. Diese erstellten XML – Dateien werden in einen bestimmten Ordner (Health/xml) gespeichert. Der Pfad hierzu ist der Selbe wie beim Uploadordner (siehe Abbildung 28).

```
class Solr
{
    //Verbindung zu SolR wird hergestellt
    function __construct() {
        //var_dump(shell_exec('java -jar Apache_SolR/solr-4.2.0/example/start.jar'));
    }

    //XML Dateien können indiziert werden
    function post() {
        include 'mainDir.php';
        if(shell_exec('java -jar ' . $mainDir . '/Apache_SolR/solr-4.2.0/example/exampledocs/post.jar ' .$mainDir. '/Health/xml/*.xml')) {
            echo 'Indizierung war erfolgreich!';
        } else {
            echo 'Fehler beim Indizieren der Dateien!';
        }

        //gepostete XML-Dateien wieder löschen
        foreach (glob($mainDir . "/Health/xml/*.xml") as $filename) {
            unlink($filename);
        }
    }

    //Indizierte Dateien können gelöscht werden mit Angabe der ID
    function delete($id) {
        file_get_contents("http://localhost:8983/solr/update?stream.body=<delete><query>id:".$id."</query></delete>&commit=true");
    }
}
```

Abbildung 37: Klasse Solr Solr.php

Mit der Klasse „Solr“ werden nun alle erstellten XML – Dateien indiziert (siehe Abbildung 37). Dies erfolgt durch den Aufruf der Methode „post()“. Alle Dateien können durch einen einzigen Befehl „*.xml“ abgearbeitet werden. Um die Indizierung zu gewährleisten, muss der Service SolR am Laufen sein. Ohne diesen Service wird die Applikation nicht funktionieren. Im Server wird das WAR File eingebunden, somit wird der Service bereitgestellt.

Damit indizierte Dateien auch wieder gelöscht werden können, steht die Methode „delete($id)“ zur Verfügung. Hier wird durch die Angabe der ID das spezifische Element gelöscht.

Neben der Indizierung werden auch die Bilder des PDFs gespeichert. Dies erfolgt über die Klasse „PDFBox“. Damit diese Klasse funktionieren kann, wird ein JAR File verwendet, welches sich im Ordner „lib“ befindet (siehe Abbildung 38).

Abbildung 38: Ordner lib JAR Files

```php
class PDFBox
{
    public $file;
    public $filename;
    public $dir;

    function __construct($file, $filename, $dir) {
        $filenameTmp = substr($filename, 0, -4);
        $this->filename = $filenameTmp;
        $this->file = $file;
        $this->dir = $dir;
    }

    //Alle Bilder eines PDF's werden extrahiert
    function getImages() {
        shell_exec('java -jar lib/pdfbox-app-1.8.2.jar ExtractImages ' . $this->file);

        if ($verzeichnis = opendir( $this->dir )) {
            $bildnamen = array();
            while (false !== ($datein = readdir($verzeichnis))) {
                $format = substr($datein, -3);
                if ($datein!="." && $datein!=".." && $format!="pdf") {
                    $bildnamen[] = $datein;
                }
            }
            closedir($verzeichnis);
        }
        $dest = $this->dir . '/' . $this->filename . '_images';
        mkdir($dest);
        foreach ($bildnamen as $bild) {
            $src = $this->dir . '/' . $bild;
            copy($src, $dest . '/' . $bild);

            unlink($src);
        }
    }
}
```

Abbildung 39: Klasse PDFBox PDFBox.php

Im Konstruktor werden die wichtigen Informationen gespeichert, welche für die Erstellung der Ordnerstruktur unerlässlich sind (siehe Abbildung 39). In der Methode „getImages()“ wird als erstes das JAR File mit dem PDF ausgeführt. Somit erhält man alle Bilder des Dokuments. Diese werden allerdings automatisch in denselben Ordner erstellt. Deshalb wird in der „while“ Schleife alle Namen der Bilder in ein Array gespeichert. Diese werden dann in einen neu erstellten Ordner kopiert. Die alten Bilder werden dann vom nicht gewünschten Ort entfernt. Somit wird eine klare Ordnerstruktur ermöglicht.

Zur Darstellung des Ladebalkens wird die Klasse „progressbar“ verwendet. Diese habe ich allerdings nicht selber geschrieben. Die Quelle für diese Klasse ist unter (fabi, 2008) erreichbar.

4.6 Update

4.6.1 GUI

Dokumente, die nicht richtig oder unvollständig indiziert wurden, können mit Hilfe von „GUIUpdate.php“ angepasst werden.

```php
<?php
include_once("klassen/SessionKontrolle.php");
include 'klassen/mainDir.php';
$id = $_GET['id'];
include 'klassen/XMLUpdate.php';
$update = new XMLUpdate($id);
include 'klassen/SolR.php';
$solr = new Solr();
//alte Informationen holen (Title, Author)
$doc = $update->getDoc();
?>
```

Abbildung 40: Includes GUIUpdate.php

Damit vor dem Laden der Seite die bereits indizierten Informationen vorhanden sind, muss zuerst eine Abfrage durchlaufen werden. Hierzu wird zuerst die ID des gewünschten Dokuments geholt (siehe Abbildung 40). Diese wird über die URL übergeben. Anschließend wird über die Klasse „XMLUpdate“ eine Abfrage an den SolR-Server geschickt. Dieser gibt ein Array zurück, welches alle Informationen über das gewünschte Dokument enthält. Dieses Array wird dann in die Variable „$doc“ gespeichert. Somit wird der Zugriff auf die bereits indizierten Informationen ermöglicht.

```
<form action="" method="post">
    <table>
        <tr>
            <td>Titel:</td>
            <td><textarea rows="3" cols="40" name="title"><?php echo $doc['response']['docs'][0]['title']?></textarea></td>
        </tr>
        <tr>
            <td>Autor:</td>
            <td><textarea rows="4" cols="40" name="author"><?php echo $doc['response']['docs'][0]['author']?></textarea></td>
        </tr>
        <tr>
            <td>Konferenz/Journal:</td>
            <td><input type="text" name="conf" value="<?php echo $doc['response']['docs'][0]['conference']?>"></td>
        </tr>
        <tr>
            <td>Ort:</td>
            <td><input type="text" name="location" value="<?php echo $doc['response']['docs'][0]['location']?>"></td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type="submit" value="speichern" name="save">
                <input type="submit" value="loeschen" name="delete">
            </td>
        </tr>
    </table>
</form>
```

Abbildung 41: HTML Code GUIUpdate.php

Im GUI selber werden vier Elemente zum Ändern angeboten (siehe Abbildung 41). Der User kann den Titel und den Autor, wo wohl die meisten Anpassungen von Nöten sein werden, zusammen mit der Konferenz/Journal und dem Land/ Ort ändern. Wenn die Informationen neu aufgesetzt worden sind, können diese, durch den Button „speichern“, gespeichert werden. Wenn allerdings das Jahr oder die Abkürzung der Konferenz/ des Journals falsch sein sollte, muss die indizierte Datei, durch den Button „löschen“, gelöscht werden. Ansonsten würde die Ordnerstruktur nicht mehr übereinstimmen.

```
<?php
if (isset($_POST['save'])) {
    //abgändertes XML speichern
    $update->save($_POST['title'], $_POST['author'], $_POST['conf'], $_POST['location']);

    //XML auf SolR indizieren
    include_once("klassen/SolR.php");
    $solr = new Solr();
    $solr->post();
}

if (isset($_POST['delete'])) {
    //abgändertes XML löschen
    $solr->delete($id);
}
?>
```

Abbildung 42: Funktionen GUIUpdate.php

Die eingefügten Elemente werden der Klasse „XMLUpdate“ übergeben und dort auch neu gespeichert (siehe Abbildung 42). Die Neuindizierung wird ausgeführt, indem ein neues XML

Dokument, allerdings mit der gleichen ID, erstellt und dieses dann dem SolR-Server übergeben wird. So wird die alte Indizierung überschrieben.

Damit eine indizierte Datei gelöscht werden kann, wird die Methode „delete“, wie in Abbildung 37, aufgerufen.

Das File „Update.css“ ist hier für die Darstellung verantwortlich.

4.6.2 Klassen

Um Indizierungen zu ändern muss ein neues XML erstellt werden. Hierzu existiert eine neue Klasse.

```
class XMLUpdate
{

    public $doc;

    function __construct($id) {
        $xmlTmp = file_get_contents("http://localhost:8983/solr/collection1/select/?indent=on&q=id:".$id."&wt=phps");
        $this->doc = unserialize($xmlTmp);
    }
```

Abbildung 43: Klasse XMLUpdate Konstruktor XMLUpdate.php

Im Konstruktor wird mit dem Übergeben der ID das bereits indizierte Dokument in ein Array gespeichert (siehe Abbildung 43). Mit diesem Array können dann alle Informationen über das gewünschte Dokument ausgegeben werden.

```
function save($title, $author, $conference, $location) {
    include 'mainDir.php';
    $xml = "<add>\n";
    $xml .= "<doc>\n";

    $xml .= "<field name=\"id\">".$this->doc['response']['docs'][0]['id']."</field>\n";
    $xml .= "<field name=\"filename\">".$this->doc['response']['docs'][0]['filename']."</field>\n";
    $xml .= "<field name=\"date\">".$this->doc['response']['docs'][0]['date']."</field>\n";
    $xml .= "<field name=\"pages\">".$this->doc['response']['docs'][0]['pages']."</field>\n";
    $xml .= "<field name=\"title\">".$title."</field>\n";
    $xml .= "<field name=\"author\">".$author."</field>\n";
    $xml .= "<field name=\"year\">".$this->doc['response']['docs'][0]['year']."</field>\n";
    $xml .= "<field name=\"conference\">".$conference."</field>\n";
    $xml .= "<field name=\"location\">".$location."</field>\n";
    $xml .= "<field name=\"conf\">".$this->doc['response']['docs'][0]['conf']."</field>\n";
    $xml .= "<field name=\"content\">".$this->doc['response']['docs'][0]['content'][0]."</field>\n";

    $xml .= "</doc>\n";
    $xml .= "</add>";

    $xml = utf8_encode($xml);


    $element = new SimpleXMLElement($xml);
    $name = substr($this->doc['response']['docs'][0]['filename'], 0, -4);
    $element->saveXML($mainDir . "/Health/xml/" . $name . ".xml");
}
```

Abbildung 44: Klasse XMLUpdate speichern XMLUpdate.php

Das schon bestehende XML wird mit den neuen Informationen ersetzt (siehe Abbildung 44). Diese werden über den Methodenaufruf übergeben. Die anderen Elemente werden von dem Array mit den alten Werten gespeichert. Hier ist sehr wichtig, dass die gleiche ID genommen wird, wie die, des schon gespeicherten Dokuments. So wird verhindert, dass eine neue Indizierung vorgenommen wird. Die alte Indizierung wird dann überschrieben.

4.7 Search

4.7.1 GUI

Die Eingabe der Suche wird in der Navigation eingebunden. Somit hat man die Möglichkeit von jeder Position der Webapplikation die Suche zu starten. Falls dies vorgenommen wird, öffnet sich das GUI „GUISearch.php“. Hier werden dann alle Dokumente, die zu dem eingegebenen Begriff passen, ausgegeben.

```php
<?php
include_once("klassen/SessionKontrolle.php");
include 'klassen/mainDir.php';
include_once("klassen/CommentdatenLesen.php");
?>
```

Abbildung 45: Includes GUISearch.php

Damit bei der Ausgabe der gefunden Dokumenten auch die durchschnittliche Bewertung der Dokumente angegeben werden können, muss die Klasse „CommentdatenLesen“ eingebunden werden (siehe Abbildung 45).

```php
<?php
    If (isset ($_POST['sucheBnt']))  {
        $art = $_POST['art'];
        $suche = $_POST['suche'];
        if ($art == "*") {
            $query = $suche;
        }  else {
            $query = $art . "%3A" . $suche;
        }
        $output = "*";

        $xmlTmp = file_get_contents("http://localhost:8983/solr/collection1/select/?indent=on&q=".$query."&wt=phps&fl=".$output);
        $xml = unserialize($xmlTmp);

        $cnt = 0;
        while ($cnt < $xml['response']['numFound'] && $cnt < 100){
            Commentdaten::commentProDoc($xml['response']['docs'][$cnt]['id']);
            $rate = 0;
            $cnt2 = 0;
            foreach (Commentdaten::$listeProDoc as $daten) {
                $rate = $rate + intval($daten->getRate());
                $cnt2++;
            }
            if ($cnt2 != 0) {
                $rate = round($rate/$cnt2);
            }  else {
                $rate = 0;
            }
```

Abbildung 46: HTML Teil 1 GUISearch.php

Auch hier werden die gefunden Dokumente in ein Array gespeichert (siehe Abbildung 46). Mit dem Element „numFound“ wird angegeben wie viele Resultate gefunden wurden. Die maximale Anzahl wurde auf 100 limitiert. Falls diese geändert werden möchte, muss zusätzlich im „config-file“ von SolR dies abgeändert werden. In der „foreach“ Schleife werden alle Bewertungen (Wertungen möglich von 1 bis 5) für das bestimmte Dokument zusammengezählt. Anschließend wird die Summe durch die Anzahl geteilt. So erhält man den Durchschnitt.

```
$target = "/library/"
        . $xml['response']['docs'][$cnt]['year'] . '/'
        . $xml['response']['docs'][$cnt]['conf'] . '/'
        . $xml['response']['docs'][$cnt]['filename'];
$link = $xml['response']['docs'][$cnt]['title'];

echo "<br/>";
echo "<a href=\"".$target."\">$link</a>" . "---" . "<a href=\"/GUIUpdate.php?id=".$xml['response']['docs'][$cnt]['id']."\">Update</a>";
echo "<br/>";
echo $xml['response']['docs'][$cnt]['author'];
echo "<br/>";
echo $xml['response']['docs'][$cnt]['content'][0];
echo "<br/>";
```

Abbildung 47: HTML Teil 2 GUISearch.php

Je Resultat wird ein Link zum Öffnen (als Titel) und zum Verändern des Dokuments eingefügt (siehe Abbildung 47). Außerdem werden die Autoren und die ersten Zeilen des Inhalts ausgegeben (siehe Abbildung 48).

TRENCADIS - A WSRF Grid MiddleWare for Managing DICOM Structured ---Update
Reporting Objects Ignacio Blanquer1, Vicente Hernandez1, DamiÃƒÂ Segrelles1
1Instituto de Aplicaciones de las TecnologÃƒÂas de
Comment (1 Comments)

Abbildung 48: Darstellung eines Dokumentes

```
echo "<a href=\"GUIComment.php?id=".$xml['response']['docs'][$cnt]['id']."\">Comment</a>" . " ";
switch ($rate) {
    case 0:
    break;
    case 1:
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star1\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star2\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star3\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star4\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star5\">";
    break;
    case 2:
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star1\">";
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star2\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star3\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star4\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star5\">";
    break;
    case 3:
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star1\">";
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star2\">";
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star3\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star4\">";
        echo "<img src=\"images/star_bw_klein.png\" id=\"Star5\">";
    break;
    case 4:
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star1\">";
        echo "<img src=\"images/star_gold_klein.png\" id=\"Star2\">";
```

Abbildung 49: HTML Teil 3 GUISearch.php

Damit die Dokumente bewerten werden können, ist zusätzlich ein Link auf „GUIComment“ enthalten. Daneben wird die durchschnittliche Bewertung, in Form von Sternen, und die Anzahl von Bewertungen angezeigt. Je nachdem was für eine durchschnittliche Bewertung zuvor ausgerechnet wurde, werden die Anzahl Sterne variieren (siehe Abbildung 49).

Auch hier wird für die Darstellung ein CSS File mit dem Namen „Search.css“ verwendet.

4.7.2 Klassen

Für die Suche wird nur die Klasse „Commentdaten“ verwendet. Diese greift auf die Datenbank zu, um alle Informationen über die Kommentare zu holen. Die genaue Funktionsweise dieser Klasse wird im nächsten Kapitel 4.8 detaillierter beschrieben.

4.8 Comment

4.8.1 GUI

In dieser GUI werden alle Kommentare eines Dokumentes angezeigt. Außerdem kann der User zusätzlich noch seinen eigenen Kommentar mit Bewertung abgeben.

```php
<?php
include_once("klassen/SessionKontrolle.php");
include 'klassen/mainDir.php';
$doc_id = $_GET['id'];
$user_id = $_SESSION['benutzer'];
//Daten aus Tabelle holen
include_once 'klassen/CommentdatenLesen.php';
include_once 'klassen/UserdatenLesen.php';
Commentdaten::commentProDoc($doc_id);
Commentdaten::commentProDocProUser($doc_id, $user_id);
//Variable zur Überprüfung, comment schon vorhanden oder nicht
$vorhanden = false;
if (is_object(Commentdaten::$listeProDocProUser)) {
    $vorhanden = true;
}
?>
```

Abbildung 50: Includes GUIComment.php

Damit diese Seite angezeigt werden kann, müssen auch hier zuerst wieder verschiedene Informationen geholt werden. Hierfür werden zwei IDs benötigt. Zum Einen die ID des Dokuments, welche wieder über die URL übergeben wird. Zum Anderen die ID des Users, damit der bereits gespeicherte Kommentar und Bewertung, falls das Dokument von diesem

User schon bewertet wurde, direkt angezeigt wird. Wenn keine Bewertungen für das Dokument vorhanden sind, wird die boolean Variable „$vorhanden“ nicht auf true gesetzt. Das hat zur Folge, dass beim Speichern der Kommentare ein „insert“ verwendet wird, ansonsten ein „update“ (siehe Abbildung 50).

```
<body onload="rating(
    <?php if (is_object(Commentdaten::$listeProDocProUser)) {
    echo Commentdaten::$listeProDocProUser->getRate();
}  else {
    echo '0';
} ?>)">
```

Abbildung 51: HTML Body GUIComment.php

Damit beim neuen Laden der Seite die Bewertung visuell richtig dargestellt wird, muss im body Tag im „onload“ die gespeicherte Bewertung, durch Aufruf der JS Methode „rating()“ gesetzt werden (siehe Abbildung 51).

```
<form action="" method="post">
    <img src="images/star_bw.png" id="Star1" onclick="rating(1)">
    <img src="images/star_bw.png" id="Star2" onclick="rating(2)">
    <img src="images/star_bw.png" id="Star3" onclick="rating(3)">
    <img src="images/star_bw.png" id="Star4" onclick="rating(4)">
    <img src="images/star_bw.png" id="Star5" onclick="rating(5)">
    <br />
    <h3>Comment</h3>
    <textarea rows="6" cols="50" name="comment"><?php if ($vorhanden) {
        echo Commentdaten::$listeProDocProUser->getComment();
    } ?></textarea>
    <br />
    <br />
    <input type="submit" value="speichern" name="save">
    <input type="hidden" name="rate" id="rat">
</form>
```

Abbildung 52: HTML Einzelner Kommentar GUIComment.php

Um ein Dokument zu bewerten, werden 5 Sterne zur Verfügung gestellt. Der User hat die Wahl wie viele Punkte er dem Dokument geben möchte. Durch klick auf die Sterne, zum Beispiel auf den vierten Stern, wird die JS Methode „rating()“ aufgerufen (siehe Abbildung 52). Diese ändert die Sterne so, dass vier Sterne goldig werden. Beim klick auf den fünften Stern werden alle Sterne goldig und so weiter.

Das versteckte Input Element „rate“ wird dazu verwendet, um auf die eingegebene Bewertung zugreifen zu können.

Zusätzlich zu der Bewertung kann das Dokument auch kommentiert werden. Falls der User für das bestimmte Dokument schon einen Kommentar gespeichert hat, wird dieser angezeigt (siehe Abbildung 53).

Abbildung 53: Darstellung Kommentar mit Bewertung Eingabe

```
<h3>Weitere Comments</h3>
<?php
    foreach (Commentdaten::$listeProDoc as $daten){
        Benutzerdaten::benutzerById($daten->getUserID());
        ?>
        <textarea rows="6" cols="50">
            <?php echo $daten->getComment(); ?>
        </textarea>
        <label>
            <?php echo "<br/>" . Benutzerdaten::$liste[0]->getVorname() . " "
                    . Benutzerdaten::$liste[0]->getName() . " "
                    . $daten->getTime() ?>
        </label>
        <br />
        <?php
        switch ($daten->getRate()) {
        case 1: ?>
            <img src="images/star_gold_klein.png" id="Star1">
            <img src="images/star_bw_klein.png" id="Star2">
            <img src="images/star_bw_klein.png" id="Star3">
            <img src="images/star_bw_klein.png" id="Star4">
            <img src="images/star_bw_klein.png" id="Star5">
        <?php break;

        case 2: ?>
            <img src="images/star_gold_klein.png" id="Star1">
            <img src="images/star_gold_klein.png" id="Star2">
```

Abbildung 54: HTML Weitere Kommentare GUIComment.php

Alle Kommentare mit Bewertungen von allen Usern werden hier angezeigt. Damit alle Bewertungen auch zugeordnet werden können, werden verschiedene Informationen zu den einzelnen Kommentaren ausgegeben. Dies sind der Vorname und der Name des Users, von welchem dieser Kommentar geschrieben wurde. Zusätzlich zu den Userinformationen wird die Zeit mit Datum angezeigt, wann dieser Kommentar gespeichert wurde (siehe Abbildung 54).

Die Bewertung wird daneben wieder in Form von Sternen angezeigt (siehe Abbildung 55).

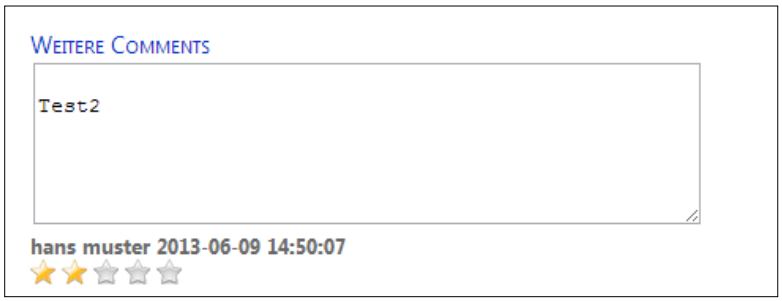

Abbildung 55: Darstellung Weitere Comments

```php
<?php
If (isset ($_POST['save']))  {
    //Daten in Tabelle speichern
    include_once 'klassen/CommentdatenSpeichern.php';
    $rate = $_POST['rate'];
    $comment = $_POST['comment'];
    $timestamp = time();
    $time = date("Y.m.d - H:i:s", $timestamp);
    if ($vorhanden) {
        $id = Commentdaten::$listeProDocProUser->getId();
        CommentVerwalten::updateComment($id, $user_id, $doc_id, $rate, $comment, $time);
    }  else {
        CommentVerwalten::addComment($user_id, $doc_id, $rate, $comment, $time);
    }
    ?>
    <script type="text/javascript">
        location.reload();
    </script>
    <?php
}
?>
```

Abbildung 56: Funktionen GUIComment.php

Beim Betätigen des Buttons „speichern“ werden die Information in die Datenbank gespeichert. Falls in der Datenbank schon ein Eintrag besteht, wird dieser angepasst, ansonsten wird ein Neuer erstellt (siehe Abbildung 56).

Anschließend wird die Seite direkt neu geladen. Dies erfolgt über JS. Somit kann der User die Änderungen direkt am Bildschirm sehen.

4.8.2 Datenbank

Damit die Kommentare für jedes Dokument gespeichert werden können, wird eine Datenbanktabelle „comments“ verwendet. Diese speichert verschiedene Informationen (siehe Abbildung 57):

#	Name	Typ	Kollation	Attribute	Null	Standard	Extra	Aktion
1	ID	int(11)			Nein	*kein(e)*	AUTO_INCREMENT	Bearbe
2	user_ID	int(11)			Nein	*kein(e)*		Bearbe
3	doc_ID	varchar(50)	latin1_swedish_ci		Nein	*kein(e)*		Bearbe
4	rate	int(11)			Nein	*kein(e)*		Bearbe
5	comment	varchar(50)	latin1_swedish_ci		Nein	*kein(e)*		Bearbe
6	time	datetime			Nein	*kein(e)*		Bearbe

Abbildung 57: Datenbank Struktur Tabelle comments

- ID: wird die Eindeutigkeit gewährleistet
- user_ID: wer hat den Kommentar geschrieben
- doc_ID: für welches Dokument ist der Kommentar
- rate: wie hoch ist die Bewertung
- comment: der Inhalt des Kommentars
- time: wann wurde der Kommentar erstellt

	ID	user_ID	doc_ID	rate	comment	time
Bearbeiten Kopieren Löschen	1	1	01-Budgen-HG05-Broker.pdf14	2	Test2	2013-06-09 14:50:07
Bearbeiten Kopieren Löschen	4	1	18-Lawford-Davies-ECIT-HG05.pdf13	4	Test	2013-06-09 14:27:23
Bearbeiten Kopieren Löschen	5	1	GRAPP_2012_100_CR.pdf5	4	Test	2013-06-10 11:45:57
Bearbeiten Kopieren Löschen	6	1	GRAPP_2012_103_CR.pdf6	4	Test	2013-06-10 11:50:53
Bearbeiten Kopieren Löschen	7	1	18500379.pdf11	4	Test2	2013-06-16 11:51:11

Abbildung 58: Datensätze Tabelle comments

So könnte dann ein Datensatz aussehen (siehe Abbildung 58). Alle wichtigen Informationen sind gespeichert und können so auch wiederverwendet werden.

4.8.3 Klassen

Um die Kommentare zu verwalten werden zwei Klassen „Commentdaten“ und „CommentVerwalten“ verwendet. Ausserdem wird für die Indizierung eine weitere Klasse „XMLUpdateComment“ verwendet.

```php
include_once 'dbConnection.php';

class Commentdaten extends DB {

    //Array f□r die Commentdaten
    static $listeProDoc = array();
    static $listeProDocProUser;


    //Funktion, um die Kommentare eines Dokuments in das oben angelegte Array zu speichern
    static function commentProDoc($id){
        $sql = 'SELECT ID, user_ID, rate, comment, time
            from comments
            where doc_ID like "' . $id . '"';
        $array = parent::getdbh()->query($sql);
        foreach ($array as $row) {
            Commentdaten::$listeProDoc[]=new Commentdaten($row['ID'], $row['user_ID'],
                    $row['rate'], $row['comment'], $row['time']);
        }
    }

    //Funktion, um die Kommentare eines Dokuments in das oben angelegte Array zu speichern
    static function commentProDocProUser($doc_id, $user_id){
        $sql = 'SELECT ID, user_ID, rate, comment, time
            from comments
            where doc_ID like "' . $doc_id . '" and
                user_ID like ' . $user_id;
        $array = parent::getdbh()->query($sql);
        foreach ($array as $row) {
            Commentdaten::$listeProDocProUser=new Commentdaten($row['ID'], $row['user_ID'],
                    $row['rate'], $row['comment'], $row['time']);
        }
    }
```

Abbildung 59: Klasse Commentdaten CommentdatenLesen.php

Die Klasse „Commentdaten“ (siehe Abbildung 59) enthält zwei Methoden. Die Methode „commentProDoc“ holt alle Kommentare für ein bestimmtes Dokument. Somit wird auch die ID übergeben.

Die zweite Methode „commentProDocProUser“ holt alle Kommentare für ein bestimmtes Dokument und von einem bestimmten User.

```php
include_once 'dbConnection.php';

class CommentVerwalten extends DB {

    //Die Daten werden in die Datenbank gespeichert
    static function addComment($user_id, $doc_id, $rate, $comment, $time){
            $sql = '
            INSERT INTO comments(ID, user_ID, doc_ID, rate, comment, time)
            VALUES (null, \''.$user_id.'\', \''.$doc_id.'\', \''.$rate.'\',
                "'.$comment.'", "'.$time.'")';

            parent::getDBH()->query($sql);
    }


    //Die Daten werden in die Datenbank geändert
    static function updateComment($id, $user_id, $doc_id, $rate, $comment, $time){
            $sql = '
            UPDATE comments
            SET user_ID=\''.$user_id.'\', doc_ID=\''.$doc_id.'\', rate=\''.$rate.'\',
                comment="'.$comment.'", time="'.$time.'"
            WHERE ID=\''.$id.'\'';

            parent::getDBH()->query($sql);
    }
}
```

Abbildung 60: Klasse CommentVerwalten CommentdatenSpeichern.php

Die zweite Klasse „CommentVerwalten“ (siehe Abbildung 60) ist dafür zuständig, die Kommentare zu speichern. Hierzu werden zwei Methoden verwendet. Falls der Kommentar noch nicht existiert hat, wird ein neuer Eintrag in die Datenbank geschrieben und somit die Methode „add()“ aufgerufen.

Wenn allerdings schon ein Kommentar von diesem User zu diesem Dokument geschrieben wurde, wird dieser nur abgeändert. Somit wird die Methode „updateComment()“ aufgerufen.

In der Klasse „XMLUpdateComment“ ist die Methode „save()“ wichtig. Sie enthält die nötigen Funktionalitäten.

```php
function save($comment, $rate, $commentOld) {
    include 'mainDir.php';
    $xml = "<add>\n";
    $xml .= "<doc>\n";

    $xml .= "<field name=\"id\">".$this->doc['response']['docs'][0]['id']."</field>\n";
    $xml .= "<field name=\"filename\">".$this->doc['response']['docs'][0]['filename']."</field>\n";
    $xml .= "<field name=\"date\">".$this->doc['response']['docs'][0]['date']."</field>\n";
    $xml .= "<field name=\"pages\">".$this->doc['response']['docs'][0]['pages']."</field>\n";
    $xml .= "<field name=\"title\">".$this->doc['response']['docs'][0]['title']."</field>\n";
    $xml .= "<field name=\"author\">".$this->doc['response']['docs'][0]['author']."</field>\n";
    $xml .= "<field name=\"year\">".$this->doc['response']['docs'][0]['year']."</field>\n";
    $xml .= "<field name=\"conference\">".$this->doc['response']['docs'][0]['conference']."</field>\n";
    $xml .= "<field name=\"location\">".$this->doc['response']['docs'][0]['location']."</field>\n";
    $xml .= "<field name=\"conf\">".$this->doc['response']['docs'][0]['conf']."</field>\n";
    $xml .= "<field name=\"content\">".$this->doc['response']['docs'][0]['content'][0]."</field>\n";
    foreach ($this->doc['response']['docs'][0]['comment'] as $com) {
        if ($commentOld!=$com) {
            $xml .= "<field name=\"comment\">".$com."</field>\n";
        }
    }
    $xml .= "<field name=\"comment\">".$comment."</field>\n";
    $xml .= "<field name=\"rate\">".$rate."</field>\n";

    $xml .= "</doc>\n";
    $xml .= "</add>";

    $xml = utf8_encode($xml);

    $element = new SimpleXMLElement($xml);
    $name = substr($this->doc['response']['docs'][0]['filename'], 0, -4);
    $element->saveXML($mainDir . "/Health/xml/" . $name . ".xml");
}
```

Abbildung 61: Klasse XMLUpdateComment XMLUpdateComment.php

Hier wird ein neues XML erstellt, welches für die Indizierung unerlässlich ist (siehe Abbildung 61). Alle Felder werden mit den schon gespeicherten Werten erstellt. Auch die Kommentare, welche nicht von mir geschrieben wurden, werden wieder gespeichert. Mein Kommentar wird nur angepasst, dies wird mit der „if-Bedingung“, welche den alten Wert

nicht mehr speichert, gewährleistet. Zusätzlich zu dem Kommentar wird die Anzahl Sterne gespeichert. Somit hat dann der User die Möglichkeit nach diesen Sternen zu suchen.

4.9 Benutzer

4.9.1 GUI

Damit auch mehrere Personen sich anmelden können, hat der User die Möglichkeit einen neuen User zu erstellen. Dies wird allerdings erst ermöglicht, wenn der Benutzer sich schon eingeloggt hat. Somit wird verhindert, dass sich jeder einfachen Zutritt verschaffen kann. Außerdem hat nicht jeder diese Berechtigung. Ausschließlich der Admin kann diese Funktion anwenden. Dies wird in der Navigation abgefragt. Wenn die Session den Namen „admin“ hat, wird die Funktion über einem Link zur Verfügung gestellt, ansonsten wird sie nicht dargestellt.

```
<form action="" method="post">
    <div class="benutzerContent">
        <div class="benutzerTable">
            <table>
                <tr>
                    <td>Benutzername:</td>
                    <td><input type="text" name="user"></td>
                </tr>
                <tr>
                    <td>Passwort:</td>
                    <td><input type="password" name="pwd"></td>
                </tr>
                <tr>
                    <td>Passwort (wdh):</td>
                    <td><input type="password" name="pwd2"></td>
                </tr>
                <tr>
                    <td>Name:</td>
                    <td><input type="text" name="name"></td>
                </tr>
                <tr>
                    <td>Vorname:</td>
                    <td><input type="text" name="vorname"></td>
                </tr>
                <tr>
                    <td>Email:</td>
                    <td><input type="text" name="email"></td>
                </tr>
            </table>
        </div>
        <div class="benutzerButton">
            <input type="submit" value="erstellen" name="erstellen">
        </div>
    </div>
</form>
```

Abbildung 62: HTML Teil GUIBenutzer.php

Pro Benutzer werden fünf Informationen gespeichert (siehe Abbildung 62). Der Benutzername und das Passwort, welche für das Login verwendet werden, der Vorname und der Name, welche bei den Kommentaren angezeigt werden und die E-Mail.

```php
<?php
If (isset ($_POST['erstellen']))  {
    include_once 'klassen/UserdatenSpeichern.php';
    $user = $_POST['user'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    $name = $_POST['name'];
    $vorname = $_POST['vorname'];
    $email = $_POST['email'];
    //Die beiden Passwörter werden miteinander verglichen
    if ($pwd == $pwd2) {
        //Wird kontrolliert ob keine Felder leer sind
        if ($user!="" && $pwd!="" && $name!="" && $vorname!="" && $email!="") {
            UserVerwalten::addUser($user, $pwd, $name, $vorname, $email);
        }  else {
            ?>
                <script type="text/javascript" language="Javascript">
                    alert("Keine Felder duerfen leer sein!")
                </script>
            <?php
        }
    }  else {
        ?>
            <script type="text/javascript" language="Javascript">
                alert("Passwort stimmt nicht ueberein!")
            </script>
        <?php
    }
}
?>
```

Abbildung 63: PHP Teil GUIBenutzer.php

Alle Funktionen werden im PHP Teil aufgerufen (siehe Abbildung 63). Zuerst werden alle eingegebenen Elemente in Variablen gespeichert. Anschließend wird kontrolliert, ob die zwei Passwörter auch übereinstimmen. Falls dies nicht der Fall sein sollte, wird eine Meldung ausgegeben, dass diese Passwörter nicht übereinstimmen. Wenn diese Bedingung erfüllt ist, wird kontrolliert, ob alle Felder ausgefüllt sind, ansonsten wird auch hier eine Meldung ausgegeben. Erst wenn beide Bedingungen erfüllt sind, wird der User über die Klasse „UserVerwalten“ gespeichert.

4.9.2 Klassen

Hierzu wird nur eine Klasse „UserVerwalten“ verwendet. Sie stellt eine Methode zur Verfügung, welche das Speichern des Benutzers ermöglicht (siehe Abbildung 64).

```php
<?php

include_once 'dbConnection.php';

class UserVerwalten extends DB {

    //Die Daten werden in die Datenbank gespeichert
    static function addUser($user, $pwd, $name, $vorname, $email){
            $sql = '
            INSERT INTO users(ID, user, pwd, name, vorname, email)
            VALUES (null, "'.$user.'", "'.$pwd.'", "'.$name.'",
                "'.$vorname.'", "'.$email.'")';

            parent::getDBH()->query($sql);
    }
}


?>
```

Abbildung 64: Klasse UserVerwalten UserdatenSpeichern.php

Mit einer simplen SQL (Structured Query Language) Abfrage wird der Benutzer in die Datenbank gespeichert. Alle wichtigen Elemente werden der Methode „addUser“ übergeben und in die Abfrage integriert.

4.10 Erweiterte Suche

4.10.1 GUI

Um eine genauere Suche der PDFs zu ermöglichen, wird eine erweiterte Suche zur Verfügung gestellt. Hier können mehrere verschiedene Angaben eingebracht werden, welche alle zusammen als Kriterien der Suche gebraucht werden.

```html
<form action="GUISearch.php" method="post">
    <div class="searchExpertContent">
        <table>
            <tr>
                <td>Zeitraum (JJJJ)</td>
                <td>
                    <input class="erwSucheJahr" type="text" name="jahrVon">
                    <t> - </t>
                    <input class="erwSucheJahr" type="text" name="jahrBis">
                </td>
            </tr>
```

Abbildung 65: HTML Teil1 GUISearchExpert.php

Zum Beispiel kann ein Zeitraum angegeben werden, wenn Dokumente zwischen 2008 und 2010 gesucht werden (siehe Abbildung 65). Falls nur das erste Feld ausgefüllt wird, wird auch nur Dokumente dessen Jahres ausgegeben. Wenn beide Felder leer bleiben, werden sie einfach vernachlässigt.

```
<tr>
    <td>Kommentar</td>
    <td><input type="text" name="comment"></td>
</tr>
<tr>
    <td>Autor</td>
    <td><input type="text" name="autor"></td>
</tr>
<tr>
    <td>Titel</td>
    <td><input type="text" name="titel"></td>
</tr>
<tr>
    <td>Ort</td>
    <td><input type="text" name="ort"></td>
</tr>
<tr>
    <td>Konferenz</td>
    <td><input type="text" name="konferenz"></td>
</tr>
```

Abbildung 66: HTML Teil2 GUISearchExpert.php

Alle anderen Felder, wie Kommentare, Titel und so weiter, können zusätzlich angegeben werden (siehe Abbildung 66). Auch hier gelten die gleichen Regeln. Die Felder die leer sind, werden nicht berücksichtigt, alle anderen werden mit einem „AND“ oder „OR“ verknüpft.

```
            <tr>
                <td>Sterne</td>
                <td>
                    <img src="images/star_bw_klein.png" id="Star1" onclick="ratingKlein(1)">
                    <img src="images/star_bw_klein.png" id="Star2" onclick="ratingKlein(2)">
                    <img src="images/star_bw_klein.png" id="Star3" onclick="ratingKlein(3)">
                    <img src="images/star_bw_klein.png" id="Star4" onclick="ratingKlein(4)">
                    <img src="images/star_bw_klein.png" id="Star5" onclick="ratingKlein(5)">
                    <input type="hidden" name="rate" id="rat">
                </td>
            </tr>
        </table>
        <div class="searchExpertButton">
            <input type="submit" value="suchen" name="erwSuche">
        </div>
    </div>
</form>
```

Abbildung 67: HTML Teil3 GUISearchExpert.php

Zusätzlich können auch nach der Anzahl Sterne gesucht werden. Diese werden standardmäßig mit den fünf Sternen gekennzeichnet (siehe Abbildung 67).

Für die Anzeige und Darstellung wird auch hier ein CSS File mit dem Namen „SearchExpert.css“ verwendet.

4.11 Mobile Login

4.11.1 GUI

Um die Webapplikation auch über ein Smartphone einfach bedienen zu können, ist zusätzlich zum normalen Browser Interface auch ein mobile Interface erstellt worden. Allerdings sind hier nicht alle Funktionen verfügbar, wie z.B. das Verändern der Indexierung, das hochladen und indizieren von Dateien und das Kommentieren und Bewerten.

```
<head>
        <title>Digital libary - Mobile</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css" />
        <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.js"></script>
</head>
<body>
        <div data-role="page">
                <!-- Container f&uuml;r Inhalte -->
                <!-- Eif&uuml;gen Header und Navigation -->
                <div id="head" data-theme="b" data-role="header">
                    <h3>
                        Digital library
                    </h3>
                </div>

                <!-- Ab hier beginnt die eigentliche seite -->

                <div data-role="content">
                    <form action="" method="post">
                        <div data-role="fieldcontain">
```

Abbildung 68: HTML GUIMobileLogin.php

Vieles hat sich zum Mobile Interface nicht geändert. Die Funktionalitäten sind genau die Selben wie beim normalen Browser Interface. Einzig an der Darstellung wurden Veränderungen angebracht. Im Head wurden scripts und ein CSS File hinzugefügt, die die Darstellung für Smartphones generieren. Außerdem wurden die div Tags mit Zusätzen, wie „data-role“ oder „data-theme“, erweitert.

Mit „data-role“ wird die Funktion des DIV’s z.B. content und mit „data-theme“ die Farbe angegeben (siehe Abbildung 68).

4.12 Mobile Suche

4.12.1 GUI

Auch bei der Suche bleiben die Funktionalitäten die Selben. Der Header wurde genau gleich verändert wie bei Abbildung 68.

```
<div data-position="left" data-display="reveal" data-theme="a">
    <ul data-role="listview" data-theme="a" data-icon="false">
        <li><a href="GUIMobileExpertSuche.php">Erweitert</a></li>
    </ul>
    <br />
</div>
<div data-role="fieldcontain">
    <select name="art" size="1">
        <option>*</option>
        <option>filename</option>
        <option>title</option>
        <option>location</option>
        <option>conference</option>
        <option>author</option>
        <option>year</option>
        <option>comment</option>
    </select>
    <br />
    <input type="text" name="suche">
    <br />
    <input type="submit" value="suchen" name="sucheBnt" data-theme="b" data-icon="forward" data-iconpos="left">
```

Abbildung 69: HTML GUIMobileSuche.php

Der Unterschied zum vorherigen Interface liegt darin, dass die Elemente, hier das „select“ Element, innerhalb eines „fieldcontain“ ist und ein Navigation zur erweiterten Suche eingefügt wurde. Dies wird durch eine listview erstellt.

Zum Button wurden auch noch Attribute, für die Farbe „data-theme“, für ein Bild „data-icon“ und für die Position des Bildes „data-iconpos“, hinzugefügt (siehe Abbildung 69).

4.13 Mobile erweiterte Suche

4.13.1 GUI

Die Funktionalitäten sind auch hier die Selben wie zuvor. Einzig an der Darstellung wurden Anpassungen vorgenommen.

```
<form action="GUIMobileSuche.php" method="post">
    <div data-position="left" data-display="reveal" data-theme="a">
        <ul data-role="listview" data-theme="a" data-icon="false">
            <li><a href="GUIMobileSuche.php">Back</a></li>
        </ul>
        <br />
    </div>
    <div data-role="fieldcontain">
        <fieldset data-role="controlgroup" data-type="horizontal" data-mini="true">
            <input type="radio" name="verknupfung" id="and" value="AND" checked="checked" />
            <label for="and">AND</label>

            <input type="radio" name="verknupfung" id="or" value="OR"  />
            <label for="or">OR</label>
        </fieldset>
```

Abbildung 70: HTML Teil1 GUIMobileExpertSuche.php

Im GUI der erweiterten Suche wird die Navigation in „back“ umgeändert. Somit kann der User wieder in die normale Suche zurückkehren. Die Radiobuttons wurden hier ein wenig verändert. Zuerst wurde ein „fieldset“ um die „inputs“ geschlossen. Dies ermöglicht die Gruppierung, wie an „data-role=controlgroup“, ersichtlich ist. Außerdem wird die mit Hilfe von „data-type“ die Elemente horizontal und mit „data-mini“ als kleine symbole dargestellt (siehe Abbildung 70).

```
<td>Sterne</td>
<td>
    <div data-role="fieldcontain">
        <select name="rate" size="1">
            <option>0</option>
            <option>1</option>
            <option>2</option>
            <option>3</option>
            <option>4</option>
            <option>5</option>
        </select>
    </div>
</td>
```

Abbildung 71: HTML Teil2 GUIMobileExpertSuche.php

Für die Sterne wird eine Dropdownliste dargestellt, in welcher sich der User zwischen einer Zahl von 0 – 5 entscheiden kann (siehe Abbildung 71).

Wenn die Suche über die erweiterte Suche gestartet wird, öffnet sich automatisch wieder die normale Suche für die Anzeige. So werden alle Eingabefelder nicht mehr angezeigt.

5 Probleme

Mit der Zeit traten immer wieder mal verschiedene Probleme auf, mit welchen ich mich auseinander setzen musste. All diese werde ich in diesem Kapitel beschreiben und aufzeigen wie ich diese gelöst habe.

5.1 Apache SolR/ Tika

Damit ich SolR und Tika gebrauchen konnte, musste ich einen Weg finden diese in PHP einzubinden. Normalerweise wird hierfür JAVA verwendet. Somit musste ich mir eine Library suchen, welche ich über PHP aufrufen konnte. Dies stellte mir zuerst einige Probleme dar, zum Einen musste ich über die Kommandozeile mit der Funktion „exec()“ die Library starten und zum Anderen musste ich die Rückgabe so gliedern, damit ich die einzelnen Werte separat speichern konnte. Dies löste ich mit String-Funktionen. So erhielt ich die einzelnen Informationen.

5.2 relevante Daten

Die erste Frage die ich mir stellte war, welche Daten brauche ich und wie erhalte ich diese. Einige Informationen konnte ich sehr einfach über den Apache Tika beschaffen. Allerdings nicht alle. Zum einen waren das Jahr und die Konferenz, welche ich nicht über den Service erhalten konnte, ein Problem. Dies löste ich, in dem der User diese Informationen selber eingibt. Somit fehlten mir nur noch der Titel und die Autoren. Zu diesen Daten kam ich, in dem ich den Inhalt mit Hilfe von Apache Tika herausfilterte und diesen dann mit String-Funktionen auseinanderschnitt, dass ich den Titel und die Autoren separat speichern konnte. Allerdings ist jedes Dokument anders und die Funktion nicht 100% fehlerfrei. Hierfür besteht die Möglichkeit, diese Werte in der Webapplikation anzupassen.

5.3 Gesamter Ordner hochladen

Ein größeres Problem stellte das Hochladen eines gesamten Ordners dar. Leider wurde dies aus Sicherheitsgründen verhindert. Somit können nur einzelne Dateien hochgeladen werden. Dies war aber nicht mein Ziel. Das Problem löste ich, in dem ich einen Ordner zur Verfügung stellte, in welchem jeder User die Daten kopieren kann, welche er indizieren möchte.

5.4 Daten zum SolR Service senden

Der Apache SolR Service arbeitet ausschließlich mit XML Dateien. Allerdings kann nicht jede beliebige XML Datei indiziert werden. Diese müssen bestimmte Knoten, wie „<add>“ und „<doc>“ aufweisen. Auch eine ID muss definiert sein, ansonsten wird die Datei nicht indiziert. An diesem Problem war ich eine Weile am Arbeiten, da ich der Meinung war, ein wohlgeformtes XML sollte indiziert werden können.

5.5 Neue Tags in SolR definieren

Damit im XML neue Tags definiert werden können und diese anschließend auch zum Suchen zur Verfügung stehen, müssen diese Tags im „schema.xml“ genau definiert werden. Alle Knoten welche hier nicht angegeben sind, können anschließend auch nicht gesucht werden.

5.6 Firewall

Die Firewall brachte auch seine Probleme mit sich. Die Applikation, welche zu Hause richtig lief, funktionierte in der Schule plötzlich nicht mehr. Leider war mir das Problem nicht sofort klar, deshalb suchte ich überall nach Fehlern wo eigentlich keine wahren. Durch Zufall kam mir der Gedanke, dass die Firewall meine Applikation im öffentlichen Netz blockieren könnte. Genau so war es dann auch. Der Fehler war dann schnell behoben.

5.7 Umlaute

Das Problem mit den Umlauten, habe ich mir in erster Linie selber zuzuschreiben. Ich habe von Anfang an, die Umlaute kontrolliert und angepasst. Allerdings war ich zu anpassungsfreudig und wandelte den Text vor der erstellen des XML doppelt um. Somit wurden die Umlaute wieder falsch angezeigt. Den Fehler fand ich dann relativ schnell und konnte ihn auch einfach beheben.

5.8 Mehrere User gleichzeitig hochladen

Zuerst habe ich nur einen Ordner für den Upload zur Verfügung gestellt. Die Überlegung, dass mehrere User gleichzeitig Daten hochladen möchten, machte ich mir zu dieser Zeit noch nicht. Durch den Dozent wurde ich auf dieses Problem aufmerksam gemacht. Die Lösung war relativ bald klar. Für jeden User musste ein spezifischer Ordner erstellt werden. Der Ordner wird mit „upload_“ und dem Usernamen angelegt. So hat jeder User seinen eigenen Ordner

und kann zu jeder Zeit Dateien indizieren. Dieser Ordner wird beim erstellen eines neuen Users automatisch erstellt.

5.9 Applikation auf dem Server

Ein großer Schritt der Arbeit war die Applikation auf dem Server zum Laufen zu bringen. Alle spezifischen Pfade mussten angepasst werden. Dies brachte so einige Schwierigkeiten mit sich. Zum Einen brauchte ich eine Weile bis ich die Ordnerstruktur des Servers verstanden habe, zum Anderen fand ich nicht alle Pfade im ersten Durchlauf und musste somit weiter Pfade, die angepasst werden mussten, suchen.

Ein weiteres Problem beim Server waren die Rechte. Wenn ich neue Ordner erstellt habe, hatte ich anschließend hierfür keine Rechte mehr. Das stellte noch ein Problem dar. Dies wurde dann vom Dozenten analysiert und behoben.

Schluss

Die Arbeit war für meine Kenntnisse sehr interessant. Viel Neues habe ich hinzugelernt. Zusammen mit dem schon Bekannten, wie PHP, habe ich die unbekannten Elemente, wie Apache SolR/ Tika, gut einbinden können. Allerdings musste ich für jeden Service eine Library finden, die ich mit PHP ausführen konnte. Dies stellte sich zuerst als ein Problem dar, da ich nicht genau wusste und auch nicht viele Informationen darüber finden konnte, wie der Service in PHP aufgerufen werden kann. Als ich den ersten Erfolg notieren konnte und den Service für meine Zwecke richtig aufgerufen habe, stellte sich das weitere Arbeiten an diesem Service nicht mehr als ein großes Problem dar.

Während der Entwicklung stieß ich auf mehrere kleinere und größere Probleme. Diese sind im Kapitel 5 einzeln beschrieben. Zum Glück konnten alle Probleme gelöst werden. Der schwierigste Punkt war der Titel und die Autoren aus dem Text hervorzuholen und diese dann zu speichern. Da ich diese Informationen nur über einen String des kompletten Textes erhalten kann, ist die Problematik schon ersichtlich. Jedes Dokument, das indiziert werden möchte ist anders als ein anderes. Somit kann die Funktion nicht immer die genauen Titel und Autoren speichern. Ich musste mich für ein Dokument entscheiden, bei welchem ich die Funktion anpassen kann. Somit ist es möglich, dass für einige Dokumente nicht der richtige Titel oder die richtigen Autoren gespeichert werden. Hier könnte vielleicht noch eine bessere Funktion zum bearbeiten des String eingebaut werden.

Einige Funktionen könnten sicherlich auch anders gelöst werden. Ein großer Punkt den ich anders machen würde, ist das Kopieren der Elemente in den FTP Server. Für PDF's gibt es zwei Varianten der Formatangabe. Einerseits die normale Angabe mit klein Buchstaben wie „.pdf“ und anderseits die Angabe mit großen Buchstaben „.PDF“. Leider war mir am Anfang nicht bewusst, dass die Dateien auch Formate mit Großbuchstaben enthalten könnten. Somit musste ich zum öffnen der Datei immer kontrollieren, ob es sich um ein Format mit Kleinbuchstaben und mit Großbuchstaben handelt. Erst dann konnte ich sicher gehen, dass der Link zur Datei auch richtig ist. Wenn ich das schon am Anfang gewusst hätte, hätte ich die Dateien in den FTP Server kopiert und direkt das Format in Kleinbuchstaben abgeändert. Somit hätte ich weniger zusätzlichen Code gebraucht.

Neben dem PHP und Apache Services war in der Arbeit auch noch ein Teil Mobile enthalten. Dieser war nicht relativ groß, da ich die gleichen Funktionen, die ich schon entwickelt habe, gebrauchen konnte und mich somit nur um das Interface kümmern musste. Außerdem wurde nur die Funktion zum Suchen der Elemente in das Mobile Interface eingebaut. Doch auch hier konnte ich neue Elemente kennenlernen.

Schlussendlich bin ich sehr froh mit meiner Wahl der Arbeit. Sie war interessant und ich konnte viel dazulernen.

VI Literaturverzeichnis

Adam, W. (Jahr unbekannt). *oxygen*. Abgerufen am 27. 4 2013 von http://www.oxygenxml.com/

Annen, O., & Schmitz, J. (14. 10 2009). *Fünf Mockup-Tools kurz vorgestellt*. Abgerufen am 27. 4 2013 von http://t3n.de/magazin/funf-mockup-tools-kurz-vorgestellt-wireframes-erstellen-224089/

Apache, F. S. (Jahr unbekannt). *Lucene*. Abgerufen am 17. 05 2013 von http://lucene.apache.org/

Apache, F. S. (Jahr unbekannt). *The Apache Software Foundation*. Abgerufen am 7. 5 2013 von http://www.apache.org/foundation/how-it-works.html#structure

Apache, S. F. (2011). *Apache Solr*. Abgerufen am 02. 07 2013 von Apache Solr: http://lucene.apache.org/solr/

Apache, S. F. (2013). *Apache Tika*. Abgerufen am 02. 07 2013 von Apache Tika: https://tika.apache.org/1.3/gettingstarted.html

Bourdon, R. (Jahr unbekannt). *WAMPSERVER*. Abgerufen am 24. 06 2013 von WAMPSERVER: http://www.wampserver.com/en/

fabi. (2008). *PHP Fortschrittsbalken / Ladebalken*. Abgerufen am 28. 05 2013 von PHP Fortschrittsbalken / Ladebalken: http://fabi.me/scripts-codes/php-fortschrittsbalken-ladebalken/

Gospodnetic, O., Hatcher, E., & Doug, C. (2004). *Lucene in Action.* United States of America: Manning Publications Co.

ICTS, A. (Jahr unbekannt). *AMRITA ICTS*. Abgerufen am 20. 07 2013 von AMRITA ICTS: http://icts.amrita.ac.in/web/technology/digital-library

Kollektiv. (28. 02 2004). *Lucene-java Wiki*. Abgerufen am 13. 5 2013 von http://wiki.apache.org/lucene-java/PoweredBy

Pakanati, S., & Chakrabarti, S. (2009). *mockingbird*. Abgerufen am 27. 4 2013 von https://gomockingbird.com/about/

Pham, M. (Jahr unbekannt). *phpDesigner 8*. Abgerufen am 27. 4 2013 von http://www.mpsoftware.dk/

Schmitz, C. (2000). *phpED*. Abgerufen am 27. 4 2013 von http://www.nusphere.com/index.htm

TechTaffy. (21. 02 2013). *Oracle Releases NetBeans IDE 7.3*. Abgerufen am 20. 07 2013 von Oracle Releases NetBeans IDE 7.3: http://www.techtaffy.com/oracle-releases-netbeans-ide-7-3/

Anhang I : Zeitmanagement

Damit die Arbeit auch richtig geplant werden kann, wurde am Anfang im Pflichtenheft eine detaillierte Aufgabenliste erstellt (siehe Tabelle 4). Somit habe ich immer einen groben Überblick wo ich stehe.

Tabelle 4: Zeitmanagement Sprints

Aufgabe	Sprint	Dauer	Priorisierung	Bemerkung
Login erstellen	1	20.05.13 – 09.06.13	sehr wichtig	benutzername und passwort
neues Profil erstellen	2	10.06.13 – 30.06.13	normal	
solr webservice funktionsweise analysieren	0	29.04.13 – 19.05.13	wichtig	
tikas xml output in solr testen	0	29.04.13 – 19.05.13	sehr wichtig	structured format
ordner in solr speichern	2	10.06.13 – 30.06.13	sehr wichtig	alle Dateien automatisch umwandeln und speichern
suche mit begriff	2	10.06.13 – 30.06.13	sehr wichtig	
erweiterte suche	3	01.07.13 – 28.07.13	wichtig	mit Filter usw.
logo erstellen	1	20.05.13 – 09.06.13	nice to have	
prototyp erstellen	1	20.05.13 – 09.06.13	sehr wichtig	GUI mit wichtigsten funktionen
apache tika in php einbinden	1	20.05.13 – 09.06.13	sehr wichtig	lib
apache solr in php einbidnen	1	20.05.13 – 09.06.13	sehr wichtig	lib
alle bereits existierenden Daten in SolR speichern		wann zeit	normal	
Dateien bewerten können	3	01.07.13 – 28.07.13	sehr wichtig	mit Sternen
Kommentare hinzufügen können	3	01.07.13 – 28.07.13	sehr wichtig	
interface wo alle kommentare ersichtlich sind	3	01.07.13 – 28.07.13	wichtig	
Pflichtenheft erstellen	0	29.04.13 – 19.05.13	sehr wichtig	
Dokumentation schreiben	alle	fortlaufend	sehr wichtig	
Mockups erstellen	0	29.04.13 – 19.05.13	wichtig	

Sprint 0: 29.04.13 bis 19.05.13
Sprint 1: 20.05.13 bis 09.06.13
Sprint 2: 10.06.13 bis 30.06.13
Sprint 3: 01.07.13 bis 28.07.13

Anhang II : Ladezeiten

Je nach Dateien können die Ladezeiten variieren. Größere Dateien beanspruchen mehr Zeit, da sie mehr Aufwand für den Service aufweisen (siehe Tabelle 5).

Tabelle 5: Ladezeiten der Indizierung

Konferenz	1 PDF	3 PDF's	10 PDF's	20 PDF's
HG 2005	5.9s	14.8s	1m 12.8s	1m 47.6s
HG2006	5.2s	12.8s	40.1s	1m 44.7s
MICCAI 2009	3.9s	8.9s	28.4s	1m 28.5s
IC3K 2010	4.7s	12.3s	40.7s	1m 16.9s
VISIGRAPP 2011	5.3s	13.6s	49.9s	1m 45.8s

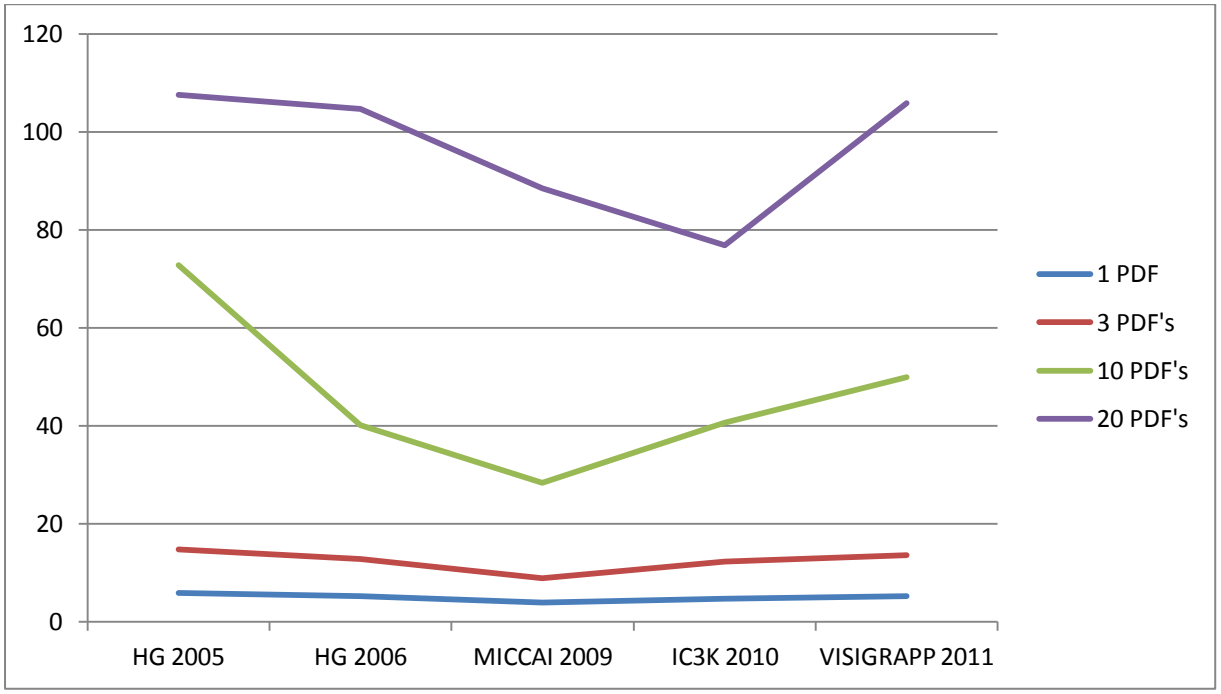

Abbildung 72: Diagramm der Ladezeiten

In der Abbildung 72 wird genau dargestellt, wie sich die Zeiten unterscheiden können. Für ein oder drei PDF's ist die Differenz noch nicht so gravierend. Allerdings bei 10 PDF's und 20 PDF's können schon größere Unterschiede ersichtlich werden.

Es ist nicht die Regel, je mehr Dateien desto grösser der Unterschied, da die Dateien unterschiedlich sortiert sein können. Das heißt, wenn die ersten 10 große und die anschließenden 10 kleine Dateien sind, wird sich die Zeit gegenüber einer Konferenz, die die umgekehrte Reihenfolge hat, wieder ausgleichen.

Selbstständigkeitserklärung

Ich bestätige hiermit, dass ich die vorliegende Bachelorarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und dass ich ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Studiengangleiters und des für die Bachelorarbeit verantwortlichen Dozenten sowie des Forschungspartners, mit dem ich zusammengearbeitet habe, werde ich diesen Bericht an niemanden verteilen, ausser an die Personen, die mir die wichtigsten Informationen für die Verfassung dieses Berichts geliefert haben und die ich nachstehend aufzähle: Prof. Henning Müller, Ivan Eggel.

Ort, Datum	Name Vorname	Unterschrift